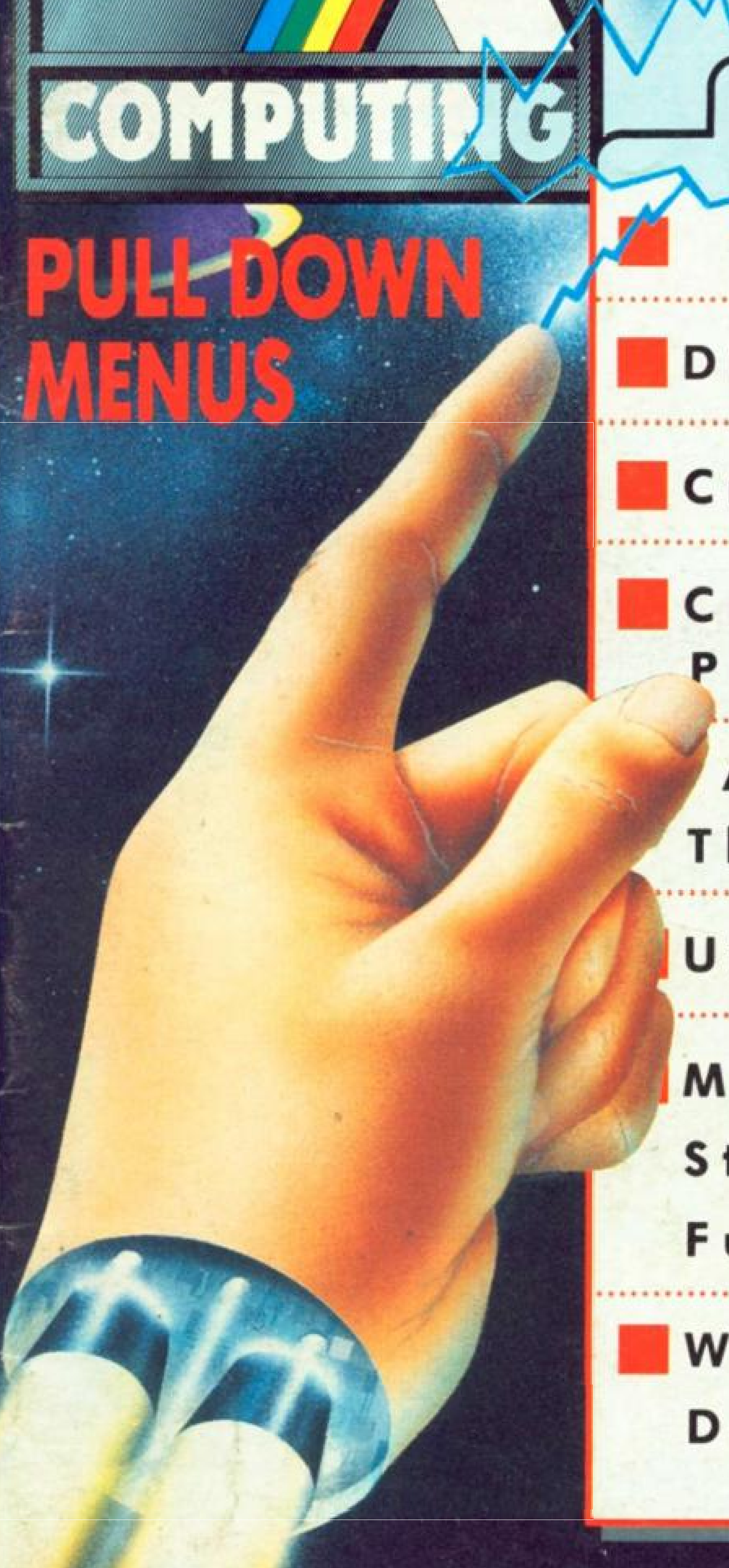
ZX COMPUTING MONTHLY

AN ARGUS SPECIALIST PUBLICATION FEBRUARY 1987 £1.50

FOR ALL SINCLAIR USERS



PULL DOWN MENUS

MENU

- Expert Systems
- Discovery Routines
- Creative Screendumps
- Crashproofing your Programs
- Advanced Art Studio/ The Artist II: Reviewed
- Utilities Revisited
- Monster Hits: Starglider: Aliens: Future Knight:
- Win: Football Fortunes: Death or Glory:



KONAMI'S COIN-OP HITS
FIVE GREAT ARCADE GAMES FOR THE PRICE OF ONE
Voted..."BEST COMPILATION OF THE YEAR"
HYPER SPORTS
Konami
GREEN BERET
Konami
Yie Ar KUNG-FU
Konami
Mikie
NOT INCLUDED FOR THE C16
Konami
PING PONG
NOT INCLUDED FOR THE BBC MICRO
Konami
KONAMI COIN-OP HITS
SPECTRUM·COMMODORE·AMSTRAD·C16·BBC
CASSETTE
£9·95
TWIN CASSETTE PACK
DISK
14·95
imagine
...the name of the game
Screen shots taken from various computer formats
Imagine Software (1984) Limited, 6 Central Street, Manchester M2 5NS. Telephone: 061-834 3939 Telex: 669977.
Distributed in France by Ocean France Telephone: 93-42-7144. Distributed in Germany by Rushware Telephone: 2101-70040

CONTENTS

REGULARS

NEWS: Opus halts Discovery Disc drive production.
4

ACROSS THE POND: U.S. news.
18

DISCOVERY COLUMN: More routines sent in by Discovery owners.
24

QL COLUMN: The latest on the QL's survival course.
35

CROSSWIRE: Readers' technical problems.
49

SHORT CUTS: Readers' prize-winning routines.
50

CROSSFIRE: Letters page.
71

RANDOM MEMORY: Programming advice from Clyde Bish.
72

PAGE 81: More from the last outpost of the 81 with Ray Elder.
81

COMPETITIONS

DEATH OR GLORY: CRL's latest space epic is up for grabs.
17

FOOTBALL FORTUNES: Win a copy of CDS's new football game.
31

PROGRAMMING PROJECTS

SPECWORD: The second part of our mega word-pro program.
42

KINGDOM OF KULL: An adventure to type in on the Spectrum.
60

PULL DOWN MENUS: Toni Baker shows how you too can add flash menus to your programs!
66

SOFTWARE REVIEWS

SPECTRUM: Aliens, Starglider, Marble Madness . . . Short reviews begin . . .
38

MINDPLAY: CRL's Dracula and The Colour of Magic among this month's releases.
56

UPGRADE ART: New versions of The Artist and Art Studio reviewed.
85

FEATURES

ILLUMINATOR: Adding decorative touches to illuminated text.
12

THE FRIENDLY PROGRAMMER: Crashproof programming by Alan Davis.
20

THAT'S THE TICKET: Carol Brooksbank on the creative use of screen dumps.
28

EXPERT SYSTEMS: Can you give your Spectrum or QL Artificial Intelliaence.
32

A BACKWARD GLANCE AT UTILITIES: Alan Davis assesses the long term performance of some established utilities.
36

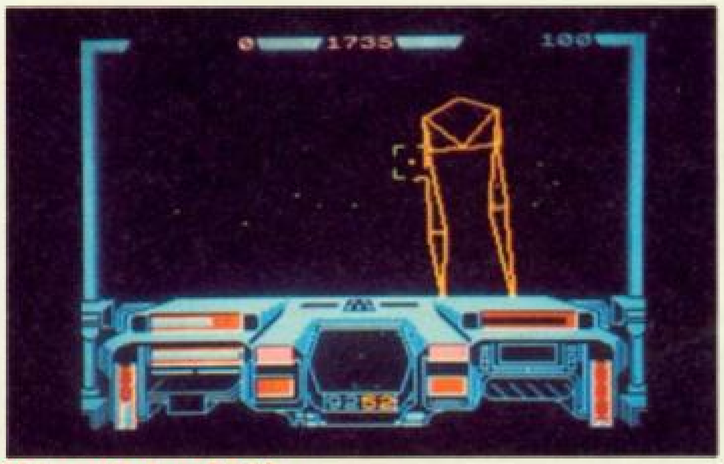
FIRST STEPS IN MACHINE CODE: Some of our regular writers tell how they started learning machine code.
52

STREAMS AND CHANNELS: Toni Baker continues to explore the windows channel.
75

TECHNICAL GRAPHICS: A look at elementary 3D graphics with Toni Baker.
78



Aliens (54)



Starglider (46)

N·E·W·S

Opus axe Discovery

Opus have ceased production of Discovery Disc Drives for the Spectrum. A spokesman for Opus said, "It's purely a commercial decision. We are now getting involved in larger more lucrative markets such as IBM and these are taking up all our available resources. We shall of course still be giving technical support to existing owners."

Opus say they have no further stocks available although some dealers may have limited supplies. An estimated 15,000 Discovery Drives have been sold since its introduction.

It is not known whether another company is to step in and continue production of the Discovery.

Dan Dare Winners

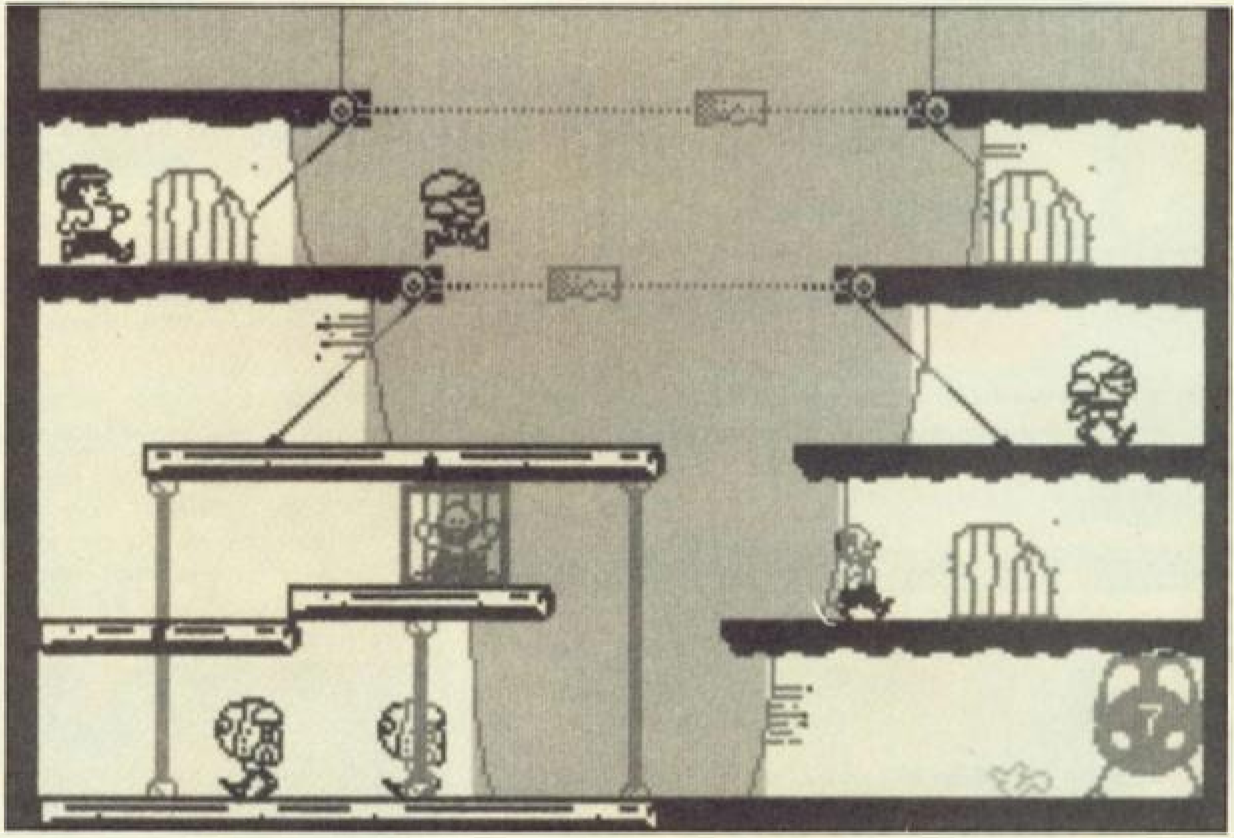
The first prize winners of the Dan Dare comptition are Maurizio Cunningham Brown from Henley in Arden and Gerry Galloway of Liverpool. They receive a copy of Virgins Dan Dare book, The Man Who Drew Tomorrow, the life story of Dan Dare's original illustrator Frank Hampson.

A further 25 entrants win a copy of the game.

They are:

R.Douglas, Mosfellssveit, Iceland; Paul Sullivan, BFPO 43; Richard Hockey, London E18; B.Atkinson, Darlington; T.Yau, Cardiff; A.Hawscroft, Manchester; D.Orunsun, Stoke; M. Kemp, Westerham; N.Almond, Coventry; E.Bennet, SW11; T.Miller, Staines; P.Dodsley, Nottingham; D.J.Morgan, Swansea; B.Herwig, Kortenburg, Belgium; A.Siddal, Chesterfield; M.Watson, Darwen, T.Witt, Galhampton; L.Voort, Leiden, Holland; C.Renders, Farnham; P. Marl, Chester; C.Womack, Northalerton; G.Shimmings, St Leonards; R.Jones, Belfast; I.McVicar, Clydebank; G.Darlington, Liverpool.

Macho Man



Advance Software are to follow up their Hardball conversion with two new titles. Indoor Sports is another conversion — a collection of Darts, Ten Pin Bowling, Blow Hockey and Ping Pong all on a single tape — and is due out in February for £8.95.

Butch Hardguy is meant to be a sort of Rambo spoof in which the aforementioned Butch has to free loads of POWs from cells on 20 different screens. The price of Butchness is £7.95.

Colossus 4 Chess winners

Chess seems to be a popular pastime among ZX readers juding from the large number of entries for our Colossus 4 Chess competition. Almost without exception every entrant deduced the correct solution to the chess problem — whites move was King C7—C8.

Now 20 winners will be able to wile away the long winter evening locked in intellectual combat with CDS's excellent Colossus 4 Chess program.

They are:

Rob Ramshaw, Tynemouth; Jim Feltham, Morden; Mark Teeger, London NW6; D.French, Margate; Mike Looseley, Harmondsworth; P.Lanff, Bad Vilbe, W.Germany; Paul Hargreaves, Brendford; De Meester Bart, Bambrugge, Belgium; John Clifton, London SE3; C.S.Evans BFPO 45; S.Deering, London E1; N.P. Powley, Kings Lynn; J.J.Carr, Cambridge; G.Havenhand, Sheffield; Brian Taylor, Scunthorpe; J.Scherphuis, Boschen Duin, The Netherlands; R. Addlesee, Leicester.

Editor: **Bryan Ralph**
Assistant Editor: **Cliff Joseph**
Consultant Editor: **Ray Elder**
Advertising Manager: **John McGarry**

Design: **Argus Design**
A.S.P. Advertising and Editorial
No. 1 Golden Square, London W1R 3AB 01-437-0626

Printed by Chase Web, Estover, Plymouth.
Advertisement Copy Controler: Andy Selwood
Distributed by: Argus Press Sales and Distribution Ltd, 12-18 Paul Street, London EC2A 4JS

ZX Computing Monthly is published on the fourth Friday of each month. Subscription rates cn be obtained from ZX Subscriptions, Infonet, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP51 1BB.

Ocean's trio

As well as their heavily hyped Christmas biggies, like Cobra and Top Gun, Ocean have also lined up a few other games with a bit less accompanying fanfare. Legend of Kage is yet another martial arts type smash 'em up, while Double Take is an odd sounding game involving cyclotrons and alternate dimensions and, of course, a healthy dose of violence. Then there's the coin-op conversion that we've all been waiting for ... Donkey Kong. Call me a cynic, but I can't help thinking they've missed the boat with that one.

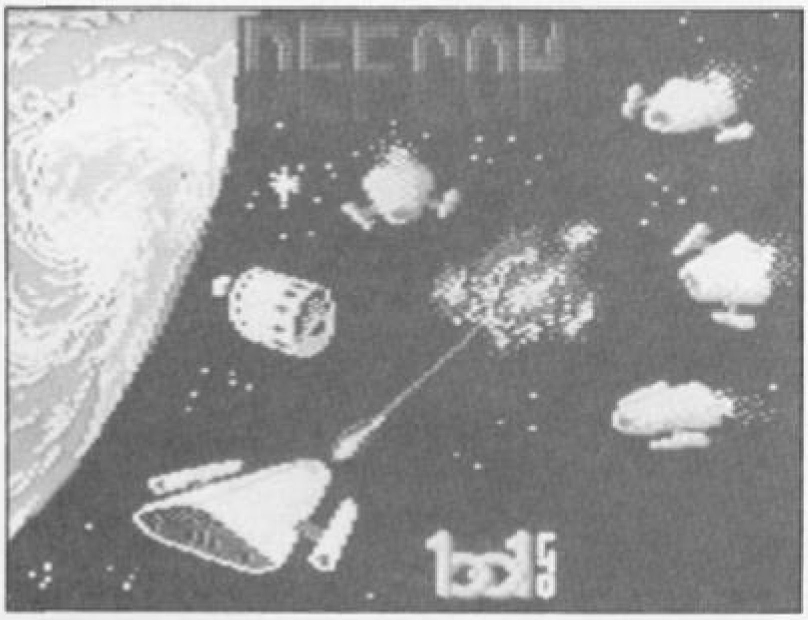
Spectrum Games Top Ten

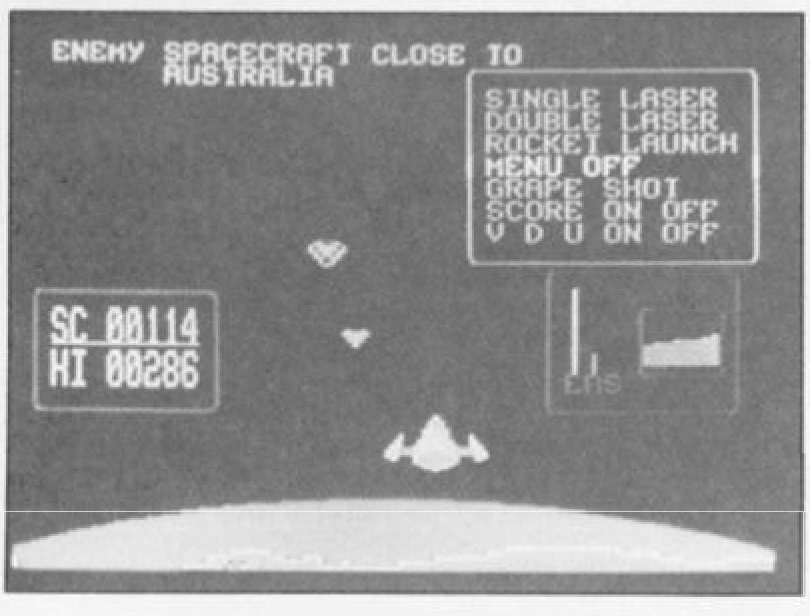
1	(7)	Trivial Pursuit	Domark
2	(1)	Computer Hits 3	Beau Jolly
3	( )	Gauntlet	US Gold
4	( )	Aliens	Electric Dreams
5	(6)	The Great Escape	Ocean
6	(3)	Paperboy	Elite
7	( )	Space Harriers	Elite
8	( )	Cobra	Ocean
9	( )	Konami's Coin OP Hits	Imagine
10	(4)	Infiltrator	US Gold

(Chart supplied by W.H. Smith)

War in Orbit

Quicksilva's latest game, Defcom, is all about the American Strategic Defence Initiative (SDI). The game has the orbiting weapons system taken over by invading aliens who decide to use it for their own purposes. Only you, in the role of heroic Nick Diamond can save the Earth, assuming that you've got £8.95 to spare to get you started.







Trivial Pursuit

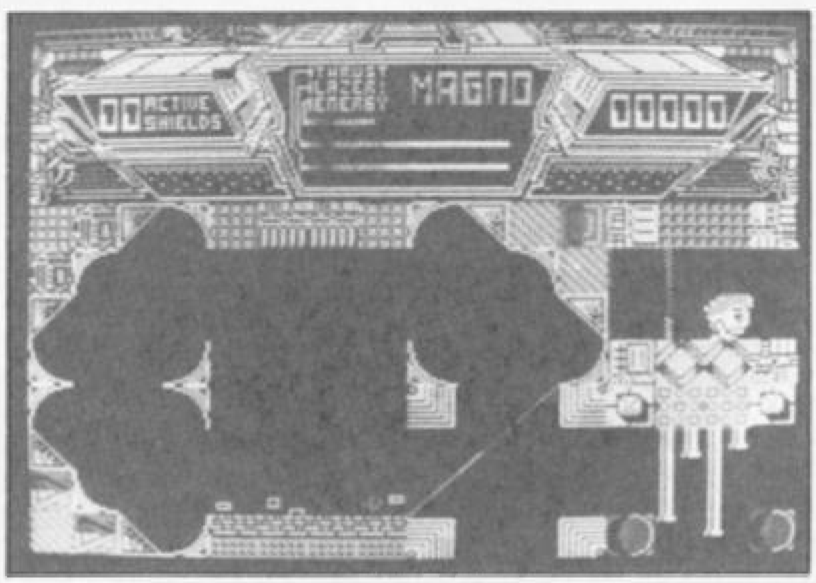
Robert Burgess of Rotherham battled bravely in the finals of Domarks Golden Trivia Challenge, held in London. As ZX's representative, Robert, narrowly missed getting into the last six by the odd wedge. Although Robert was not destined to carry off the £10,000 solid gold Trivial Pursuit Set he did take home the new Genus 2 Edition of the game as a runners up prize.

ZX BMX

We probably shouldn't admit it but the Commodore version of BMX Simulator released by budget software house Codemasters was one of our recent lunchtime favourites. Imagine then the breathless anticipation with which we await the arrival of the Spectrum version. With a track full of ramps, banks, whoops and burns there are all sorts of possibilities for cycle mayhem and all you need to join in is £1.99.

Sounds familiar

Mastertronic have got a new game lined up called Terminus. It's a massive arcade adventure with lots and lots of screens and you have to guide your cute little sprite past lots of traps and aliens and (stop me if you've heard this before...)



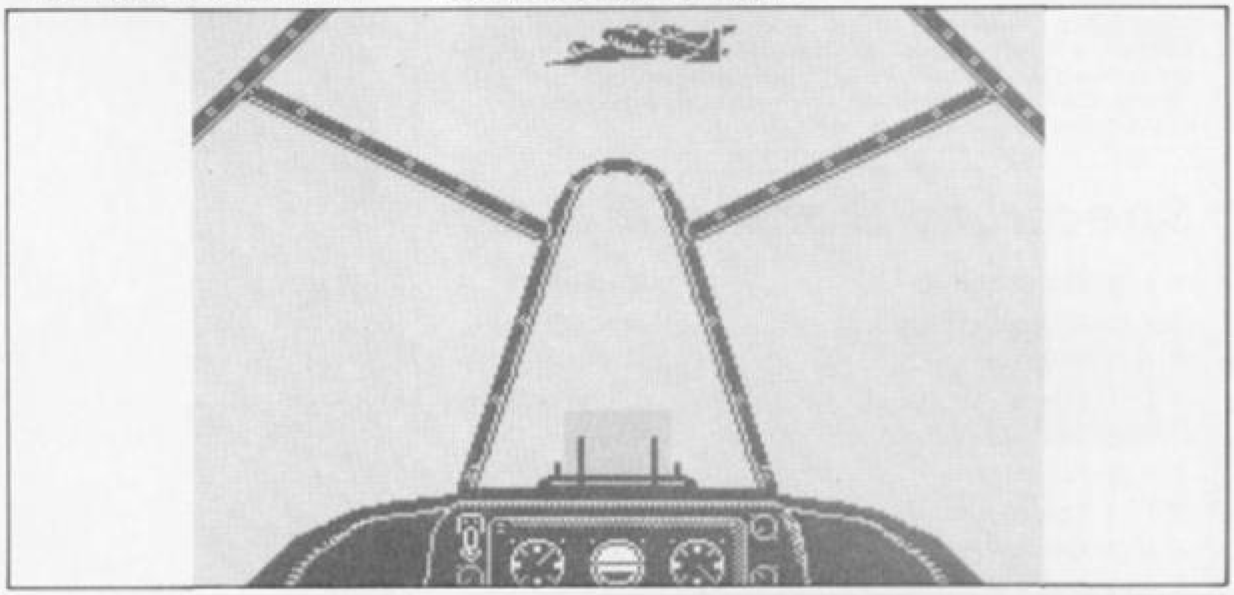
The Alternative Budget

Yet another budget software label has launched itself onto the market, in the shape of Alternative Software. Their first two offerings, priced at £1.99 are Howzat! (a cricket game, would you believe) and Henry's Hoard, a 50 screen platform game. We haven't actually seen the games yet, but the cassette inlays look nice. . .



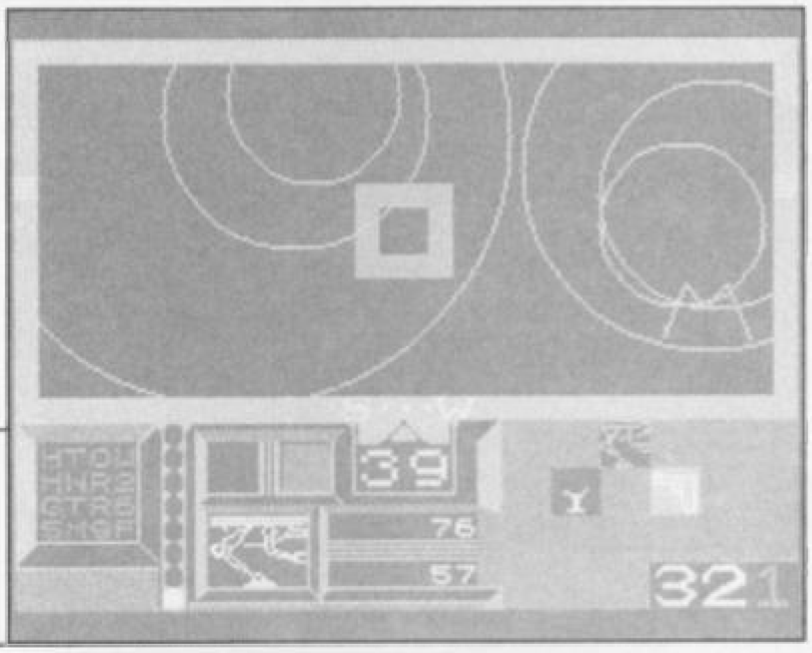
Battle of Britain

The PSS Wargamers series marches ever onwards with the release of Battle of Britain. This manages to combine a 'deploy your forces against the deadly Hun' strategy game with a few flight simulator style arcade sequences (of the 'blow the deadly Hun out of the sky' type), so it may well appeal to more than just the usual wargame following.



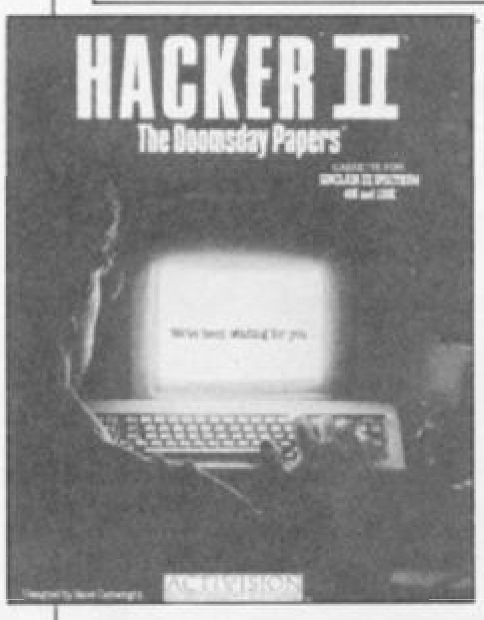
Round the bend

Okay, hands up all the overgrown kids who used to have a Scalextric kit? Well you can indulge in some computerised nostalgia with Scalextric on your Spectrum, courtesy of Leisure Genius/Virgin. the game costs £9.95



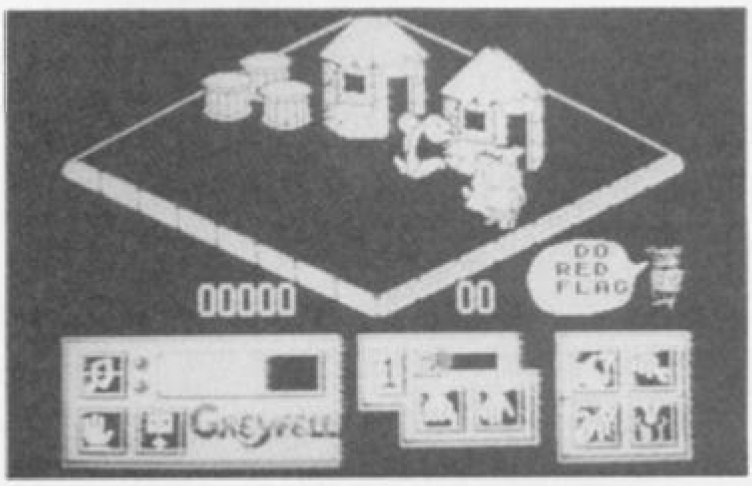
Into the hive

The Hive is a little something that Firebird are preparing for the New Year. Set inside a hive where your task is to destroy the Queen of the Hive. The game is being written by the Torus team (of Gyron fame).



Starlight

Greyfell — The Legend of Norman, is the first release by a new full price software house called Starlight. Norman is a humble cat in a world entirely inhabited by animal characters such as Potbellius, the dog landlord, Blotto, the drunk rabbit and Willy the pig policeman. Dissatisfied with his uneventful life Norman sets off to defeat the evil Moron, a ruler who has been making life hell for the denizens of the menagerie kingdom.



All this and Hacker two!

The Doomsday papers is the subtitle of the sequel to Activision's earlier Hacker game. This time you're playing for really big stakes as the CIA has asked you to save the entire planet from the Russians. You've got to do a bit of computerised spying and hack your way into the computer in a Siberian security complex.

The Doomsday Papers retails at £9. 95.

More war

Yet another WWII battle zone gets the wargame treatment. CCS have come up with Vulcan, which reproduces the Tunisian campaign of World War II. The game is for 1 or 2 players and allows you to control either Allied or Axis forces and comes packed with all sorts of maps and photos, and a big video box for £9.95.

24 HOUR

COMPUTER REPAIRS and COMPUTER SPARES

SINCLAIR

QUALITY APPROVED REPAIR CENTRE

AMSTRAD · SINCLAIR · SEAL OF QUALITY · AMSTRAD APPROVED REPAIR CENTRE

HOW TO GET YOUR SPECTRUM REPAIRED FOR ONLY £19.95



SPECIAL OFFER!

Why not upgrade your ordinary Spectrum into the fantastic DK'Tronics typewriter keyboard for only **£31.50** including fitting, VAT and return post and packing. Normal recommended retail price £49.95. Replacement printed DK'Tronics key sets **£7.50** including post & packing.

SPECIAL OFFER!

Why not upgrade your 16K Spectrum to a 48K for as little as **£17.95** including VAT, post and packing. If you would like us to fit the kit for you, just send us **£19.95** which is a fully inclusive price to cover all costs including return postage. Full fitting instructions supplied with every kit. Issue 2 and 3 only.

16K to 48K UPGRADE KIT

Update Your Rubber Keyboard to a New Spectrum+

Fitted for only **£31.90** + £1.50 post & packing (Also D.I.Y. Kit available for only **£24.95** + £1.50 post & packing)

Your Spectrum repaired and upgraded to a Spectrum Plus for special offer price of **£50.00** complete.

Same day service LIMITED OFFER



TEN ★ REPAIR SERVICE

- While you wait service including computer spare parts over the counter.
- All computers fully overhauled and fully tested before return.
- Fully insured for the return journey.
- Fixed low price of **£19.95** including post, packing and VAT. (Not a between price of really up to **£30.00** which some of our competitors are quoting).
- Discounts for schools and colleges.
- Five top games worth **£39.00** for you to enjoy and play with every Spectrum repair.
- We repair Commodore 64's, Vic 20's, Commodore 16's and Plus 4's.
- The most up to date test equipment developed by us to fully test and find all faults within your computer.
- Keyboard repairs, Spectrum rubber key boards only **£8.95**.
- 3 month written guarantee on all repairs.

TRADE REPAIRS most welcome

BEST SERVICE – BEST PRICES!

TRADE ORDERS WELCOME

EXTENSION RIBBON

56 Way ribbon cable to extend your ports for your peripherals **£10.95** plus £1.50 p & p

KEYBOARD TEMPLATES £5.50 plus £1.50 p & p

RUBBER MAT £6.50 plus £1.50 p & p

ARE YOU ANOTHER CUSTOMER - fed up waiting weeks for your estimate?

Need your computer repaired fast? Then send it now to the **Number One Repair Company in the U.K.**, or call in and see us at our fully equipped 2,500 square foot workshop, with all the **latest test equipment available**. You are more than welcome. We will repair your computer while you wait and help you with any of your technical problems. Commodore computers repaired for only **£35.00**. Please note we gove you a 100% **low fixed price of £19.95** which includes return post and packing, VAT, not a between price like some other Repair Companies offer. We don't ask you to send a cheque in for the maximum amount and **shock you** with repair bills **£30** upwards. Don't forget we are Amstrad aproved for quality and speed, don't risk your computer to any other unauthorised repair centre. We dont just repair the fault and send your computer back, we give your computer a:-

FREE OVERHAUL WITH EVERY REPAIR WE DO:-

We correct **Colour, sound, Keyboard, Check the loading and saving chip**, Put new feet on the base if required, Check for full memory, check all sockets including ear/mike and **replace where needed**. All for **an inclusive price of £19.95 including VAT, all parts, insurance and post and packing. No hidden extras whatsoever**. We don't have to boast too much about our service as we have thousands of customers from all over the world highly delighted with our service. A **first class reputation for speed and accuracy**. Don't forget, we also now have a Service Branch in **Manchester City Centre** for while you wait service.

NEW

SPECIAL OFFER!

Gun Shot II Rapid Fire Joystick and interface complete outfit. Normal recommended retail price £19.95, if purchased together. Special offer of only **£16.50** including p & p

Joystick available as separate item **£10.95** plus interface available as separate item **£9.00** plus £1.50 p & p

REPLACEMENT POWER SUPPLY

Spectrum replacement power transformer suitable for all makes of computer £9.95 plus £1.50 p & p

ZX81 SPARES NOW IN STOCK

Spectrum keyboard membranes **£5.50** plus £1.50 p & p

Spectrum Plus spare keyboard membranes **£12.90** plus £1.50 p & p

ZX 81 membranes **£5.00** plus £1.50 p & p

KEYBOARD MEMBRANES

COMMODORE POWER SUPPLY

Commodore replacement power transformer £29.00 plus £1.50 p & p

VideoVault D.I.Y. CORNER

We regret we cannot show all the components available. Just give us a call and we can quote you over the phone, delivery by 1st class post.

SPECTRUM SPARES

Z80B CPU	3.00
4116 Rams	1.00
ZTX 650	0.60
ZTX 213	0.60
Power Supply Transformers	9.95
ULA 6C001	16.50
Rom	16.50
Keyboard membrane Spectrum	5.50
Keyboard membrane Spectrum Plus membrane	12.90
Metal Templates	5.50
Keyboard Mats	5.50
ZX81 membrane	5.00
Service Manual	£30.00

COMMODORE SPARES

6526 – C.I.A.	19.00
6510 – Processor	19.00
6581 – Sid Chip	19.00
906114 – House Keeper	19.00
901225 – Graphic Rom	19.00
901226 – Basic Rom	19.00
901227 – Kernal Rom	19.00
6569 – VIC	19.00
4164 Rams – Memory	3.00
Power Supply Transformers	29.00

All our prices include VAT, but please add £1.50 on each order to cover post, packing and handling charges.

VIDEOVAULT HEALTH WARNING!!!
Sending your computer to any other Repair Centre can seriously Damage its Health

FULLY REPAIRED AND TESTED ON MOST ADVANCED TEST EQUIPMENT IN EUROPE!

VideoVault Ltd.

140 High Street West, Glossop, Derbyshire SK13 8HJ
Tel: 04574-66555/67761 Head office & access orders, queeries,
Manchester 061- 236 0376 while you wait repair centre only.

ORDER NOW!

OPEN 7 DAYS A WEEK
MANCHESTER MON - SAT ONLY

SPECIAL OFFER

5 GREAT FREE GAMES NORMAL RECOMMENDED RETAIL PRICE £39.00 YOURS FREE WITH EVERY SPECTRUM REPAIR

DANDY

Arcade action in the necromancers dungeon as the magical flak flies. . .

Dandy
Electric Dreams
£9.95

Fifteen dungeons packed with spectres, necromancers and assorted nasties lie in wait for those brave enought to enter.

Taking the role of Thor, who can be joined by Sheba (controlled by a second player) you must hack and zap your way through rooms, passageways and stairways to get to the treasure.

Dandy may seem an odd name for a massive, magical, arcade adventure but it is named after an original game that later became known as Gauntlet.

Only a fraction of each dungeon level is shown on the sceen at any one time which flip to the next section when you move off the edge of the screen.

This lack of gradual scrolling means that you often rush into situations you'd rather avoid like a horde of necromancers.

Immediately below the dungeon display is a scroll indicating the present levels of energy and number of keys, treasure and spells for each player.

You begin the game with 1000 energy units that are drained at an alarming rate whenever a nasty gets near you. Luckily this can be topped up by collecting piles of food that are strewn around the dungeon (usually on the wrong side of an army of dungeon denizens).

Keys, magic and treasure can also be found and are essential to your survival.

The keys are used to open the doors that would otherwise block your path but since there are less keys than doors, care must be taken to use them only on the doors that are important and lead to treasure or the way out. Quite often a room has several doors all leading to dead ends when the correct route is through a teleport pad that jumps you to a similar pad in an adjacent room.

Use the valuable keys on dead end doors and you won't have enough to reach the stairs that lead to the next level.

Pressing the fire button hurls a hail of blasts at the nasties in your line of fire. One hit is enough to take out most dungeon dwellers but the necromancers need 4 hits to kill them. The worst to shift are the spectres not only because they need more hits to kill them but also they can drift through walls that block the others. If you don't get them, they'll certainly get you.

There can be as many as thirty or more critters coming at you so even one shot nasties become a big problem. The answer lies in magic.

Unfortunately, the spells that you find have a random effect when they are cast. Sometimes a spell may wipe out a screenful of nasties but others may only stun them for a short time or just disorienatate them.

Even if you've wiped them out you must move quickly as they'll quickly be replaced as more are generated.

If things get really tough and your energy plummets you can trade treasure for energy that might keep you going long enough to find some more food.

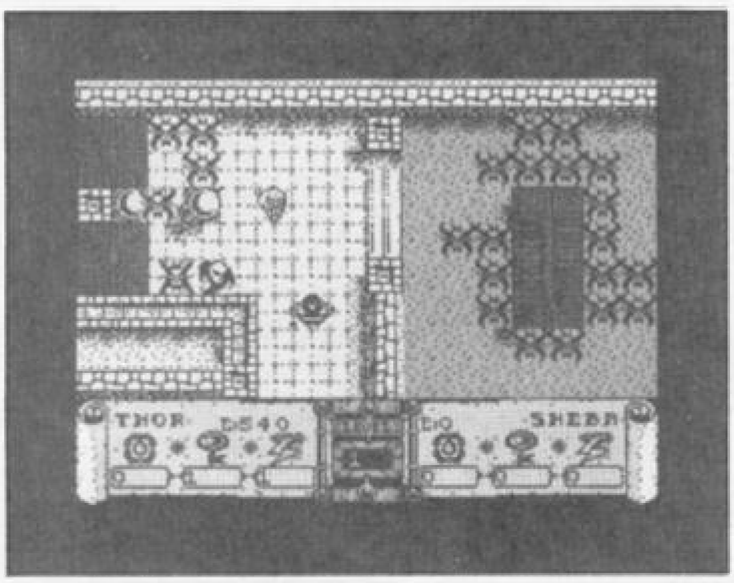
Dandy must be played at a frantic pace otherwise you will be constantly overrun with nasties. This will mean that you will make mistakes such as using a key on the door you were trying to avoid letting loose a horde of monsters that begin to chase you.

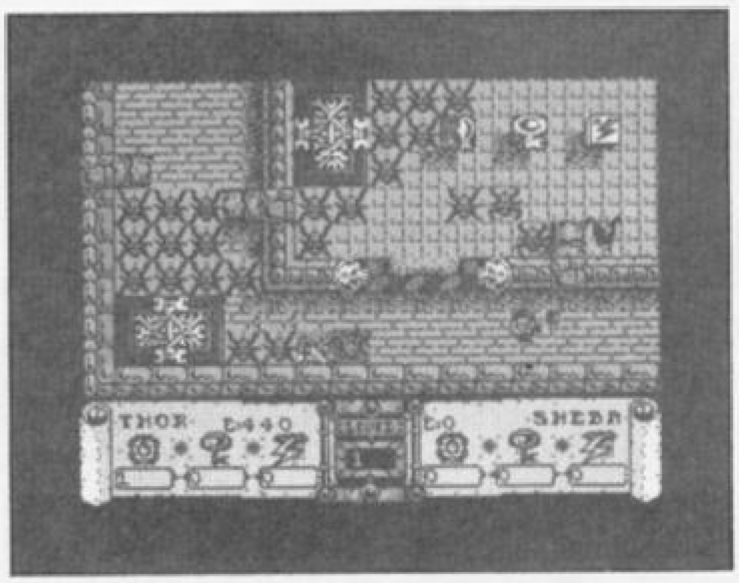
Complete a set of dungeons and you'll be awarded a clue but you'll need to survive all of them to get all three clues to solve the game.

I'm not sure of the point of these clues that are thrown in almost as an afterthought but perhaps when the riddle is solved it will make more sense. But before then I've got a few more spectres to trash!



GREAT





Four top notch games for the price of one from Durell

BIG 4

Big 4
Durell
£9.95

Four of Durell's best known games are now available in a single twin cassette pack. Now for the price of one game you can fly a deadly Lynx helicopter in Combat Lynx, drive a Turbo Esprit in a city centre car chase, infiltrate an enemy security base in Saboteur and disable an anti-matter plant in Critical Mass.

Combat Lynx

The action begins on the launch pad as you arm your combat Lynx helicopter for the mission ahead. Your job is to protect and ferry troops between a maximum of six bases (depends on game level) while fighting the planes, helicopters, tanks and gun emplacements of the enemy forces.

Through your controls you must plot the positions of the bases and the approaching forces and defend the ones most at risk while keeping the others fully supplied.

This isn't an easy game to learn with over 30 key controls to perfect but it is still one of the best combat flight simulators.

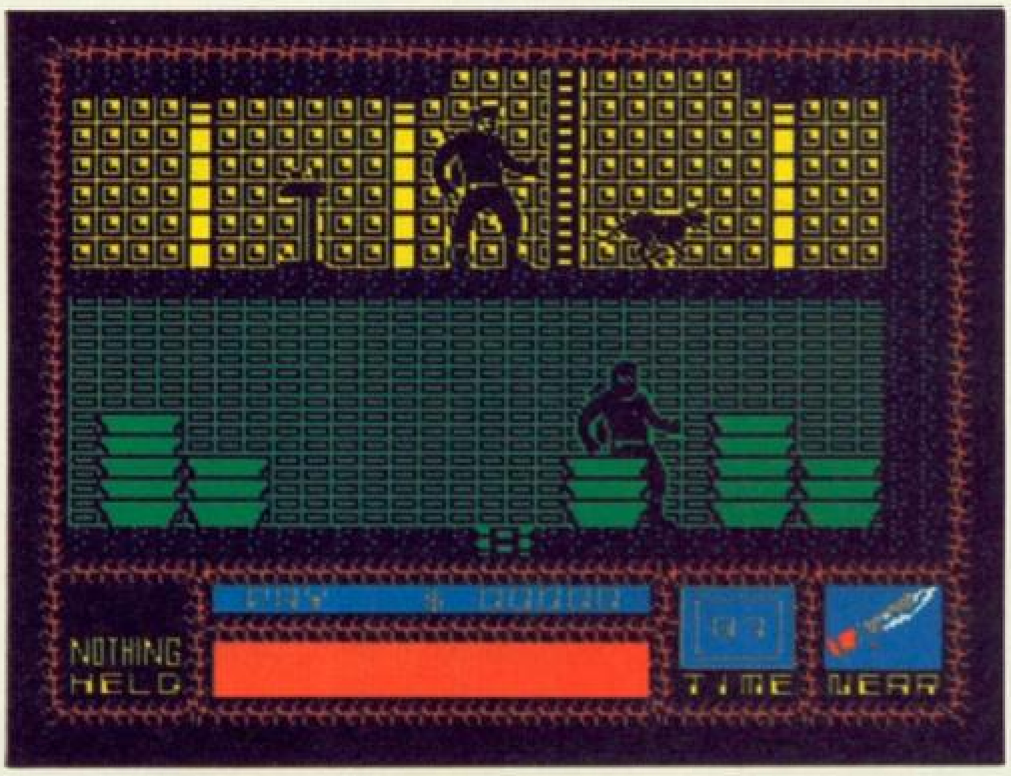
Turbo Esprit

Driving your Lotus Turbo Esprit around one of four city centres at 150mph isn't easy especially when you're supposed to be the good guy and avoid mowing down pedestrians and other drivers. Meanwhile the bad guys are operating a drugs ring and you must find and catch the armoured supply car and the four pick-up cars before the hit cars find you.

Unfortunately, you must stop at traffic lights and observe other driving laws while the drug dealers will shoot anyone in an attempt to get away.

Tracking down your targets is easy using the scanner that reports their position which you can follow on your map but look out for warnings about hit cars approaching you. These try to gun you down from behind so if you get a warning you'll need to perform a speedy manoeuvre to get behind them!

An excellent car chase game but don't be surprised if your penalties (for crashing, running lights, shooting innocent people) are greater than your score.



Saboteur

As a change of pace Saboteur has you creeping around a warehouse that the villains are using as a central security station. Your mission as an ace mercenary is to infiltrate the warehouse and find a disk that contains the names of all the rebel leaders.

Naturally you're an expert martial artist as the game was released when kung fu games ruled.

Now it is looking a little dated but has survived mainly due to the size of the warehouse you must explore and the choice of weapons you can find and use on the patrolling guards. The guards also have dogs that constantly snap at your heels and drain your energy.

Eventually you should make your way to the roof where a helicopter waits to rescue you.

Critical Mass

My pick of the bunch is Critical Mass that gives you just ten hours gametime to travel through the five zones to reach the power plant before it explodes.

Naturally this isn't going to be easy as the enemy that invaded the planet and caused all the trouble attempt to destroy your rocket propelled hovercraft. This is protected by a force field that is weakened by any collisions with the rocks that strewn the surface or by enemy fire. If this gives way your ship dramatically explodes around you leaving you hovering above a pile of rubble.

If you're lucky you can hover to a replacement pod and get another ship to continue your mission. If you're unlucky you'll be eaten by one of the Dune style giant worms that rear out of the planet to chomp you.

Should you manage to reach the base you then have to find a way in past the fused mines, disorientation clouds and protective wall before you can have a shot at the energy concentrator to clos down the reactor.

Each game separately is well worth playing with; my favourite being Combat Lynx and Critical Mass but with four hits for the price of one, it just has to be a monster hit.

MONSTER HIT


DEEP STRIKE
R.R.P. £9.95
Spectrum & Amstrad
available November 20th
Commodore 64
available January 20th
Trade enquiries to Centresoft
on 021 356 3388
Sales dept.,
Castle Lodge, Castle Green,
Taunton, Somerset TA1 4AB
England
Telephone (0823) 54489 & 54029
DURELL
software getting harder . . .



THANATOS
Spectrum in October, Amstrad in November, Commodore 64 in December.
All £9.95
DURELL
Sales dept.,
Castle Lodge, Castle Green,
Taunton, Somerset TA1 4AB
England
Telephone (0823) 54489 & 54029
software getting harder . . . .



Illuminator at work, showing one possible form of decoration

Exploring the possibilities of the Illuminator program with Alan Davies



Illuminator

Welcome back to ZX Computing's Anglo-Saxon department. If you've been dabbling with the Illuminator program from last month's article, you should now be the possessor of the following items: (a) 26 large initial letter shapes stored in 832 bytes; (b) a redesigned "normal" character set stored in 768 bytes; and very probably (c) spots before the eyes! Never mind — this month we'll make all your tribulations worthwhile, so put yourself in a suitably medieval frame of mind, and let's get down to the serious business of churning out an illuminated masterpiece or two...

What we need, of course, is **Listing 1** — this is the assembler program which we'll be using to print our strings of text. If you don't have an assembler, you can use **Listing 2** instead. This BASIC program will poke in the code for you and save it to microdrive — but you can change line 50 to an ordinary SAVE command if you're working with tape. The code is stored from 64800 onwards, and is 518 bytes long. Before you can use it, you need both the bytes for the illuminated capitals, and those for your redesigned "normal" characters residing in memory from 62976 onwards, and 64000 onwards, respectively (i.e. exactly as saved by the Illuminator program last month.) Don't forget to lower RAMTOP before loading in the three sections of code; CLEAR 62975 is what you want. (When all the various parts are in memory, you'll probably find it convenient to save a copy of the whole lot together as a single code block of 2342 bytes from 62976 onwards.)

To try it out, enter the following command:-
CLS: LET z$="Any old bit of text will do as long as the first letter is in upper case.": LET m=USR 64800.



Screen dump of demo program

Did it work? (If it didn't, you'll need to go back and check carefully through your saved copy one byte at a time, comparing it with Listing 2.) By the way, if perchance the first letter of z$ is in *lower* case you won't get a crash — you'll just get a large square of rubbish printed on screen where the illuminated capital should be.

Text effects

What else can it do for us? Well, quite a lot. There are several addresses which can usefully be poked to produce a variety of effects, and these are as follows: (the labels correspond to those in the assembly listing.)

First, there are three addresses whose contents govern the attributes of the illuminated capitals:-
65229 (BRTC) can be POKEd with 1 or 0 to change the BRIGHT attribute (normally zero).
65230 (INKC) can be POKEd with any number 0-7 to set the INK.
65231 (PAPC) can be POKEd with any number 0-7 to set the PAPER.

Then there are two addresses which set left and right margins:-
65232 (TAB) contains the width of the left hand margin.
65233 (TAB2) contains the width of the right hand margin.

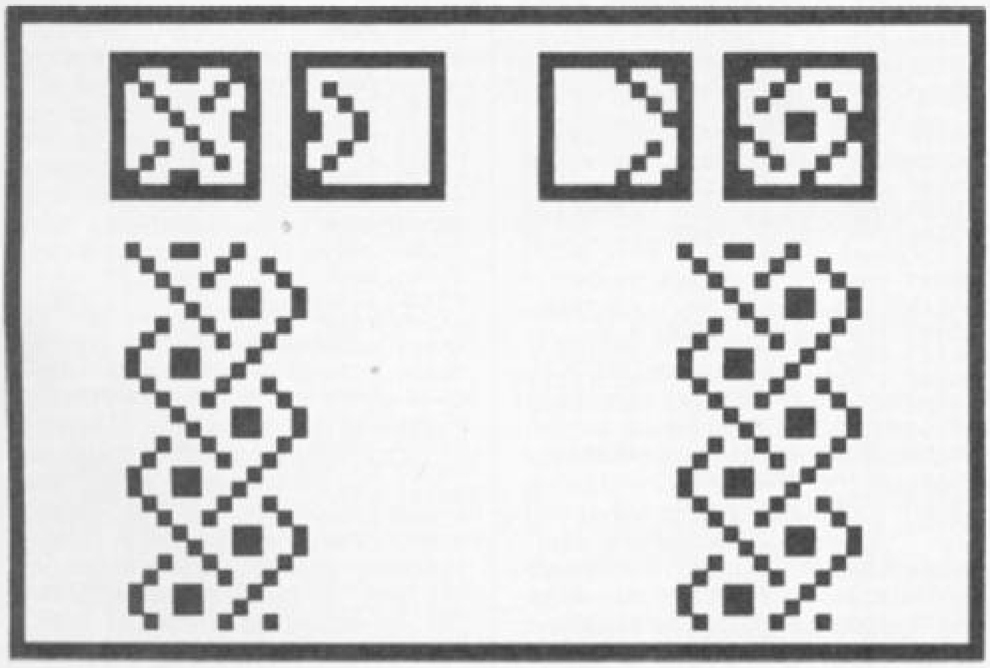
So if, for instance, you POKE 65232,2: POKE 65233,1 then your text will be printed with a maximum line length of 29 characters, inset 2 character squares from the left, and leaving a one character square margin on the right. This gives great power to your illuminating elbow, because it means you can set up a decorated border of any width on both sides of your text, and the text will *not* overprint it.

There's one address whose contents control the printing mode:-
65234 (ILLUM) can be POKEd with either 0 or 1 where 1 corresponds to illuminated capital printing, and 0 gives "normal" printing, i.e. the illuminated initials are switched off.

Flexibility

Is that all? No, there's more (after all, I did promise you a utility that was *flexible!*) Lurking among the code is a simple but effective "window" clearing facility which can be called at USR 65250. This will clear the screen between any two specified lines, and you control it using these addresses:-
65235 (TOP) contains the number of the top line to be cleared.
65236 (BOT) contains the number of the bottom line to be cleared.

So if, for example, you want to clear a window between lines 1 and 12 inclusive, then **POKE**


Figure 1

Listing 1

```
*HISOFT GENS3M2 ASSEMBLER*
    ZX SPECTRUM
Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: 00

                10 *D+
       20 *C-
64800  30            ORG  64800
       40 INIT
64800  50            XOR  A
64801  60            LD   (DONE),A
64804  70            LD   HL,63744
64807  80            LD   (CHRS),HL
64810  90            LD   A,(TAB)
64813  100           LD   HL,TAB2
64816  110           ADD  A,(HL)
64817  120           LD   B,A
64818  130           LD   A,29
64820  140           SUB  B
64821  150           LD   (LINE),A
64824  160           LD   A,1
64826  170           LD   (ONE),A
64829  180           LD   A,2
64831  190           CALL #1601
64834  200           LD   HL,(VARS)
       210 NEXT
64837  220           LD   A,(HL)
64838  230           CP   90
64840  240           JP   Z,FND1
64843  250           CALL SEARCH
64846  260           EX   DE,HL
64847  270           JP   NEXT
       280 FND1
64850  290           LD   A,(ILLUM)
64853  300           CP   0
64855  310           JP   Z,FND2
64858  320           INC  HL
64859  330           LD   A,(HL)
64860  340           DEC  A
64861  350           LD   (LEN),A
64864  360           INC  HL
64865  370           INC  HL
64866  380           LD   A,(HL)
64867  390           LD   (CAP),A
64870  400           INC  HL
64871  410           LD   (STRT),HL
64874  420           LD   A,(CAP)
64877  430           SUB  65
64879  440           CP   23
64881  450           CALL NC,SWAP
64884  460           ADD  A,A
64885  470           ADD  A,A
64886  480           ADD  A,32
64888  490           LD   (PART),A
64891  500           CALL INSET
64894  510           CALL PCAP
64897  520           CALL PCAP
64900  530           CALL SCAN
64903  540           CALL INSET
64906  550           CALL PCAP
64909  560           CALL PCAP
64912  570           LD   A,(DONE)
64915  580           CP   1
64917  590           CALL Z,NEWLN
64920  600           CALL SCAN
       610 NORM
64923  620           XOR  A
64924  630           LD   (ONE),A
64927  640           LD   A,(LINE)
64930  650           ADD  A,2
64932  660           LD   (LINE),A
64935  670           CALL SCAN
64938  680           LD   HL,62720
64941  690           LD   (TCRS),HL
64944  700           RET
       710 FND2
64945  720           INC  HL
64946  730           LD   A,(HL)
64947  740           LD   (LEN),A
64950  750           INC  HL
64951  760           INC  HL
64952  770           LD   (STRT),HL
64955  780           LD   HL,63744
64958  790           LD   (CHRS),HL
64961  800           JP   NORM
       810 SWAP
64964  820           SUB  24
64966  830           LD   HL,63488
64969  840           LD   (TCRS),HL
64972  850           RET
       860 PCAP
64973  870           LD   HL,(TCRS)
64976  880           LD   (CHRS),HL
64979  890           LD   A,(PFLG)
64982  900           RES  0,A
64984  910           LD   (PFLG),A
64987  920           LD   A,19
64989  930           RST  16
64990  940           LD   A,(BRTC)
64993  950           RST  16
64994  960           LD   A,16
64996  970           RST  16
64997  980           LD   A,(INKC)
65000  990           RST  16
65001  1000          LD   A,17
65003  1010          RST  16
65004  1020          LD   A,(PAPC)
65007  1030          RST  16
65008  1040          LD   A,(PART)
65011  1050          RST  16
65012  1060          LD   HL,63744
65015  1070          LD   (CHRS),HL
65018  1080          LD   A,(PART)
65021  1090          INC  A
65022  1100          LD   (PART),A
65025  1110          RET
       1120 SCAN
65026  1130          LD   A,(DONE)
65029  1140          CP   1
65031  1150          RET  Z
65032  1160          LD   A,(ONE)
65035  1170          CP   0
65037  1180          CALL Z,INSET
65040  1190          CALL CHECK
65043  1200          CP   1
65045  1210          JP   Z,PRINT
65048  1220          CALL CHOP
65051  1230          CALL PRINT
65054  1240          LD   HL,(STRT)
65057  1250          LD   DE,(STEP)
65061  1260          LD   D,0
65063  1270          ADD  HL,DE
65064  1280          LD   (STRT),HL
65067  1290          LD   A,(STEP)
65070  1300          LD   B,A
65071  1310          LD   A,(LEN)
65074  1320          SUB  B
65075  1330          LD   (LEN),A
       1340 REPT
65078  1350          LD   HL,(STRT)
65081  1360          LD   A,(HL)
65082  1370          CP   32
65084  1380          JP   NZ,MORE?
65087  1390          INC  HL
65088  1400          LD   (STRT),HL
65091  1410          LD   A,(LEN)
65094  1420          DEC  A
65095  1430          LD   (LEN),A
65098  1440          JP   REPT
       1450 MORE?
65101  1460          LD   A,(ONE)
65104  1470          CP   1
65106  1480          RET  Z
65107  1490          JP   SCAN
       1500 CHECK
65110  1510          LD   A,(LINE)
65113  1520          ADD  A,2
65115  1530          LD   B,A
65116  1540          LD   A,(LEN)
65119  1550          CP   B
65120  1560          JP   C,SHORT
65123  1570          XOR  A
65124  1580          RET
       1590 SHORT
65125  1600          LD   A,(LEN)
65128  1610          LD   (STEP),A
65131  1620          LD   A,1
65133  1630          LD   (DONE),A
65136  1640          RET
       1650 PRINT
65137  1660          CALL TEMPS
65140  1670          LD   A,(PFLG)
65143  1680          RES  0,A
65145  1690          LD   (PFLG),A
65148  1700          LD   DE,(STRT)
65152  1710          LD   BC,(STEP)
65156  1720          LD   B,0
65158  1730          CALL #203C
65161  1740 NEWLN    LD   A,13
65163  1750          RST  16
65164  1760          RET
       1770 CHOP
65165  1780          LD   HL,(STRT)
65168  1790          LD   DE,(LINE)
65172  1800          ADD  HL,DE
       1810 LOOP
65173  1820          LD   A,(HL)
65174  1830          CP   31
65176  1840          JP   NC,SPACE
       1850 END
65179  1860          INC  DE
65180  1870          LD   A,E
65181  1880          LD   (STEP),A
65184  1890          RET
       1900 SPACE
65185  1910          INC  HL
65186  1920          LD   A,(HL)
65187  1930          CP   32
65189  1940          JP   NZ,DECR
65192  1950          JP   END
       1960 DECR
65195  1970          DEC  DE
65196  1980          DEC  HL
65197  1990          DEC  HL
65198  2000          JP   LOOP
       2010 INSET
65201  2020          LD   A,(TAB)
65204  2030 AGAIN    CP   0
65206  2040          RET  Z
       2050
65207  2060          DEC  A
65208  2070          PUSH AF
65209  2080          LD   A,(PFLG)
65212  2090          SET  0,A
65214  2100          LD   (PFLG),A
65217  2110          LD   A,255
65219  2120          LD   (MSKT),A
65222  2130          LD   A,32
65224  2140          RST  16
65225  2150          POP  AF
65226  2160          JP   AGAIN
23627  2170 VARS     EQU  #5C4B
23606  2180 CHRS     EQU  23606
 6584  2190 SEARCH   EQU  #19B8
23697  2200 PFLG     EQU  23697
23696  2210 MSKT     EQU  23696
 3405  2220 TEMPS    EQU  #0D4D
65229  2230 BRTC     DEFB 0
65230  2240 INKC     DEFB 0
65231  2250 PAPC     DEFB 7
65232  2260 TAB      DEFB 0
65233  2270 TAB2     DEFB 0
65234  2280 ILLUM    DEFB 1
65235  2290 TOP      DEFB 0
65236  2300 BOT      DEFB 21
65237  2310 LEN      DEFB 0
65238  2320 STEP     DEFB 0
65239  2330 STRT     DEFW 0
65241  2340 CAP      DEFB 0
65242  2350 TCRS     DEFW 62720
65244  2360 PART     DEFB 0
65245  2370 LINE     DEFW 0
65247  2380 ONE      DEFB 0
65248  2390 DONE     DEFB 0
65249  2400 NUM      DEFB 0
       2410 CLEAR
65250  2420          LD   A,(BOT)
65253  2430          INC  A
65254  2440          LD   (BOT),A
65257  2450          LD   A,2
65259  2460          CALL #1601
65262  2470          CALL TEMPS
65265  2480          LD   A,32
65267  2490          LD   HL,TAB
65270  2500          SUB  (HL)
65271  2510          LD   HL,TAB2
65274  2520          SUB  (HL)
65275  2530          LD   (NUM),A
65278  2540          LD   A,(TOP)
       2550 LOOPC
65281  2560          PUSH AF
65282  2570          LD   A,22
65284  2580          RST  16
65285  2590          POP  AF
65286  2600          PUSH AF
65287  2610          RST  16
65288  2620          LD   A,(TAB)
65291  2630          RST  16
65292  2640          LD   A,(NUM)
65295  2650          LD   B,A
65296  2660 CONT     LD   A,32
65298  2670          RST  16
65299  2680          DJNZ CONT
65301  2690          POP  AF
65302  2700          INC  A
65303  2710          LD   HL,BOT
65306  2720          CP   (HL)
65307  2730          JP   C,LOOPC
65310  2740          LD   A,(BOT)
65313  2750          DEC  A
65314  2760          LD   (BOT),A
65317  2770          RET

Pass 2 errors: 00

Table used:   546 from   600
```

65235,1: POKE 65236,12: LET m=USR 65250 will do it. Note that the routine takes into account your left and right hand margins (set by TAB and TAB2) and consequently clears *only* the space within them.

Just one more point, concerning colours. The INK and PAPER for both the main text printing and screen clearing are established by whatever *permanent* INK and PAPER you set from BASIC. No other action on your part is needed.

You're now in a position to print more or less what you like, where you like, however you like — and then rub it all out again. The only limitation on what you print is that LEN z$ must be less than 255 — though of course there's nothing to prevent you from printing longer chunks of text provided you do it in bits, calling the routine to print each chunk separately.

So much for the bread and butter; now for the jam. The point of this exercise, you'll recall, is to try to produce an effect similar to illuminated manuscript which could be used to improve the presentation of a text adventure. It's fairly obvious, I think, that the effectiveness of the idea will largely depend on the decoration you put *around* the text, in addition to the text and initials themselves. There are many possible approaches to this, and the illustrations scattered around this and last month's article may give you a few ideas to get you started.

Decoration

One possibility which seems promising is simply to make the TV screen *look* like an old piece of parchment. Half an hour's work with Melbourne Draw (or similar utility) will provide a suitably "ragged" edge to the screen — and you can then load this in as a SCREEN$ and print your text onto it (POKEing appropriate values for left and right margins before you start.) The BASIC "CLS" command must be avoided of course, as it would wipe out all your decoration, but that's no problem since you can do all your screen clearing selectively using the USR 65250 call.

Another approach, either instead of or in addition to the above, is to make use of the fact that many items in the full character set are not likely to be needed, and to redefine these as suitably decorative shapes for use either alone or in combination. If you look back at Figure 1 in last month's article, for example, you'll see that I redesigned CHR$ 91-93, and CHR$ 123-125 to produce a "scroll" effect when they're printed in sequence, which makes an attractive way of dividing blocks of text.

If you're a stickler for authenticity, and want to try to mimic some of the features found on actual Anglo Saxon manuscripts, you might like to try the poor man's version of the "knotted tracery" type of decoration which I used in one of the illustrations here, and which is shown in enlarged detail in **Figure 1**. (This interweaving line motif — or variations of it — is very commonly used in Anglo-Saxon illumination.) All you need to do is design 4 characters to the shapes enclosed in boxes in Figure 1 — I chose CHR$ 94/95 and 126/127 for this. Then just PRINT CHR$ 94; CHR$ 95 all the way down the left hand side of the screen, and CHR$ 126; CHR$ 127 down the right hand side — but don't forget to set both margins 2 character squares wide. (I found it desirable to

Listing 2

```
 10 DATA 175,50,224,254,33,0,249,34,54,92
 20 DATA 58,208,254,33,209,254,134,71,62,29
 30 DATA 144,50,221,254,62,1,50,223,254,62
 40 DATA 2,205,1,22,42,75,92,126,254,90
 50 DATA 202,82,253,205,184,25,235,195,69,253
 60 DATA 58,210,254,254,0,202,177,253,35,126
 70 DATA 61,50,213,254,35,35,126,50,217,254
 80 DATA 35,34,215,254,58,217,254,214,65,254
 90 DATA 23,212,196,253,135,135,198,32,50,220
100 DATA 254,205,177,254,205,205,253,205,205,253
110 DATA 205,2,254,205,177,254,205,205,253,205
120 DATA 205,253,58,224,254,254,1,204,137,254
130 DATA 205,2,254,175,50,223,254,58,221,254
140 DATA 198,2,50,221,254,205,2,254,33,0
150 DATA 245,34,218,254,201,35,126,50,213,254
160 DATA 35,35,34,215,254,33,0,249,34,54
170 DATA 92,195,155,253,214,24,33,0,248,34
180 DATA 218,254,201,42,218,254,34,54,92,58
190 DATA 145,92,203,135,50,145,92,62,19,215
200 DATA 58,205,254,215,62,16,215,58,206,254
210 DATA 215,62,17,215,58,207,254,215,58,220
220 DATA 254,215,33,0,249,34,54,92,58,220
230 DATA 254,60,50,220,254,201,58,224,254,254
240 DATA 1,200,58,223,254,254,0,204,177,254
250 DATA 205,86,254,254,1,202,113,254,205,141
260 DATA 254,205,113,254,42,215,254,237,91,214
270 DATA 254,22,0,25,34,215,254,58,214,254
280 DATA 71,58,213,254,144,50,213,254,42,215
290 DATA 254,126,254,32,194,77,254,35,34,215
300 DATA 254,58,213,254,61,50,213,254,195,54
310 DATA 254,58,223,254,254,1,200,195,2,254
320 DATA 58,221,254,198,2,71,58,213,254,184
330 DATA 218,101,254,175,201,58,213,254,50,214
340 DATA 254,62,1,50,224,254,201,205,77,13
350 DATA 58,145,92,203,135,50,145,92,237,91
360 DATA 215,254,237,75,214,254,6,0,205,60
370 DATA 32,62,13,215,201,42,215,254,237,91
380 DATA 221,254,25,126,254,31,210,161,254,19
390 DATA 123,50,214,254,201,35,126,254,32,194
400 DATA 171,254,195,155,254,27,43,43,195,149
410 DATA 254,58,208,254,254,0,200,61,245,58
420 DATA 145,92,203,199,50,145,92,62,255,50
430 DATA 144,92,62,32,215,241,195,180,254,0
440 DATA 0,7,0,0,1,0,21,0,0,0
450 DATA 0,0,0,245,0,0,0,0,0,0
460 DATA 58,212,254,60,50,212,254,62,2,205
470 DATA 1,22,205,77,13,62,32,33,208,254
480 DATA 150,33,209,254,150,50,225,254,58,211
490 DATA 254,245,62,22,215,241,245,215,58,208
500 DATA 254,215,58,225,254,71,62,32,215,16
510 DATA 251,241,60,33,212,254,190,218,1,255
520 DATA 58,212,254,61,50,212,254,201,0,0
600>CLEAR 64799:LET s=0:RESTORE
610 FOR i=64800 TO 65317
620 READ x: POKE i,x: LET s=s+x
630 NEXT i
640 IF s<>74487 THEN  PRINT AT 10,10;"ERROR!!": STOP
650 SAVE *"m";1;"illum"CODE 64800,518
```

SPECTRUM PROGRAMMING

Listing 3

```
   7 REM
   8 REM *** CLEAR TOP WINDOW ***
   9 REM
  10 POKE 65235,1: POKE 65236,12: LET M=USR 65250: PRINT AT 1,0;
  11 RETURN
  17 REM
  18 REM *** CLEAR BOTTOM WINDOW ***
  19 REM
  20 POKE 65235,14: POKE 65236,20: LET M=USR 65250: PRINT AT 14,
0;
  21 RETURN
  47 REM
  48 REM *** PRINT Z$ WITH ILLUMINATED INITIAL ***
  49 REM
  50 LET inkc=2+INT (4*RND): POKE 65234,1: POKE 65230,inkc: POKE
 65231,papc
  51 LET m=USR 64800
  52 RETURN
  57 REM
  58 REM *** PRINT Z$ WITH NORMAL INITIAL ***
  59 REM
  60 POKE 65234,0: INK 5: LET M=USR 64800: INK 6
  61 RETURN
7997 REM
7998 REM *** LOAD M/C AND LETTER SHAPES ***
7999 REM
8000 CLEAR 59999: LOAD *"m";1;"illum"CODE : LOAD *"m";1;"chars:a
"CODE : LOAD *"m";1;"capitals:a"CODE
8007 REM
8008 REM *** PERMANENT INK/PAPER ***
8009 REM
8010 PAPER 0: INK 6: BORDER 0: CLS
8018 REM *** LEFT/RIGHT BORDERS 2 CHR SQUARES WIDE ***
8020 POKE 65232,2: POKE 65233,2
8027 REM
8028 REM *** INK/PAPER FOR INITIALS ***
8029 REM
8030 LET inkc=2: LET papc=0
8037 REM
8038 REM *** PERMANENT SCREEN DECORATION ***
8039 REM
8040 LET x$=CHR$ 16+CHR$ 4+CHR$ 64+CHR$ 16+CHR$ 3+CHR$ 91+CHR$ 9
2+CHR$ 93+CHR$ 123+CHR$ 124+CHR$ 125+CHR$ 16+CHR$ 4+CHR$ 64
8050 POKE 23606,0: POKE 23607,249: FOR i=0 TO 21: PRINT  INK 5;C
HR$ 94;CHR$ 95;AT i,30;CHR$ 126;CHR$ 127: NEXT i: PRINT AT 0,12;
x$;AT 0,3;x$;AT 0,21;x$;AT 13,3;x$;AT 13,12;x$;AT 13,21;x$;AT 21
,3;x$;AT 21,12;x$;AT 21,21;x$
8057 REM
8058 REM *** DEMONSTRATION ***
8059 REM
8060 LET z$="This is the top window, for location descriptions.
It occupies the screen from row one to row twelve inclusive. The
 decorated border is two character squares wide on each side."
8070 GO SUB 10: GO SUB 50: GO SUB 9000
8080 LET z$="Each fresh printing up here is done with the initia
l capital illuminated. You need to check that your text is not t
oo long of course!"
8090 GO SUB 50: GO SUB 9000
8100 LET z$="This is the bottom window for printing new informat
ion as the adventure proceeds."
8110 GO SUB 20: GO SUB 60: GO SUB 9000
8120 LET z$="It occupies rows fourteen to twenty inclusive. Note
 that capitals are normal, though of course they need not be."
8130 GO SUB 60: GO SUB 9000
8140 GO SUB 10: GO SUB 9000
8150 GO SUB 20: GO SUB 9000
8160 GO TO 8060
9000 PRINT #1;AT 1,10; FLASH 1;"PRESS A KEY": PAUSE 0: PRINT #1;
AT 1,0,,: RETURN
```

have a decent blank gap — half a character square — between the border decoration and the text, or things begin to look rather cluttered.)

Windows

If you're going to use this as a display method for a text adventure, then you'll need to operate with several "windows" — one for location descriptions, one for program responses, and another (perhaps the bottom two lines) for displaying the player's input. The most convenient approach is to incorporate the machine code calls within a small number of BASIC subroutines to define your windows, set the print position etc. — and I've offered some help here in the shape of **Listing 3**. If you type this in and save it to auto-run from line 8000, it will give you a simple demonstration of the effects that can be achieved.

The four relevant subroutines in Listing 3 are as follows:

GOSUB 10: This sets top and bottom limits of the upper window (screen lines 1-12 inclusive), clears it, and resets the PRINT position to the start of line 1.

GOSUB 20: This sets top and bottom limits of the lower window (screen lines 14-20 inclusive), clears it, and resets the PRINT position to the start of line 14.

GOSUB 50: This chooses a random INK colour for the illuminated capitals, selects "Illuminated initial" mode, and prints the text held in z$ at the current PRINT position.

GOSUB 60: This selects "normal" printing of z$ and shows how the colour of the text can be changed by altering the permanent INK colour. Again, printing is done from the current PRINT position.

By arranging things in this way it becomes possible to do anything you like within the two windows. If you use GOSUB 50 or GOSUB 60 alone, printing will begin on the line following the last line printed, so that several successive strings of text can be printed within the same window. Alternatively, by preceeding the text printing with a GOSUB 10 or GOSUB 20, you can clear out a window and reset the PRINT position within it.

Lines 8040/50, by the way, set up the screen decoration and will only produce a sensible display if you've defined your character set to include the "scroll" and "knotted tracery" motifs (see Figure 1 in last month's article). If you haven't, just replace those CHR$ numbers above 90 by CHR$ 42 (asterisk). The result won't look very pretty, but will still enable you to run the demonstration.

Once you've seen what the demonstration does — following the listing so that you see *why* it does it — you're all set. If you're thinking of writing an adventure, rather than just using it as a general display facility, then you might like to be reminded that the machine code and data are positioned in memory so that you can, if you wish, use the Venturespeak command analyser (see the October-December issues of ZXC). Otherwise it's over to you, and all that remains is for me to say that I hope you find the experience...... illuminating?

BULLDOZING AT THE SPEED OF LIGHT!

COMPETITION

There are thirty copies of CRL's new destruction derby in space to be won

Death or Glory from CRL is not so much a shoot 'em up as a smash 'em up. You take the controls of a space 'dozer which keeps the interstellar highways clear of meteorites and cosmic debris. But when an invading alien fleet threatens your home planet you are called upon to do battle. As your space 'dozer is unarmed your only option is to ram your enemies into submission.

Cosmic quiz

All you have to do to get your hands on a copy of Death or Glory is answer three simple out-of-this-world questions.

1) When did man first set foot on the moon?

a) 1971
b) 1973
c) 1969

2) Mercury is the nearest planet to the sun, Venus is the second nearest. Which is the third nearest?

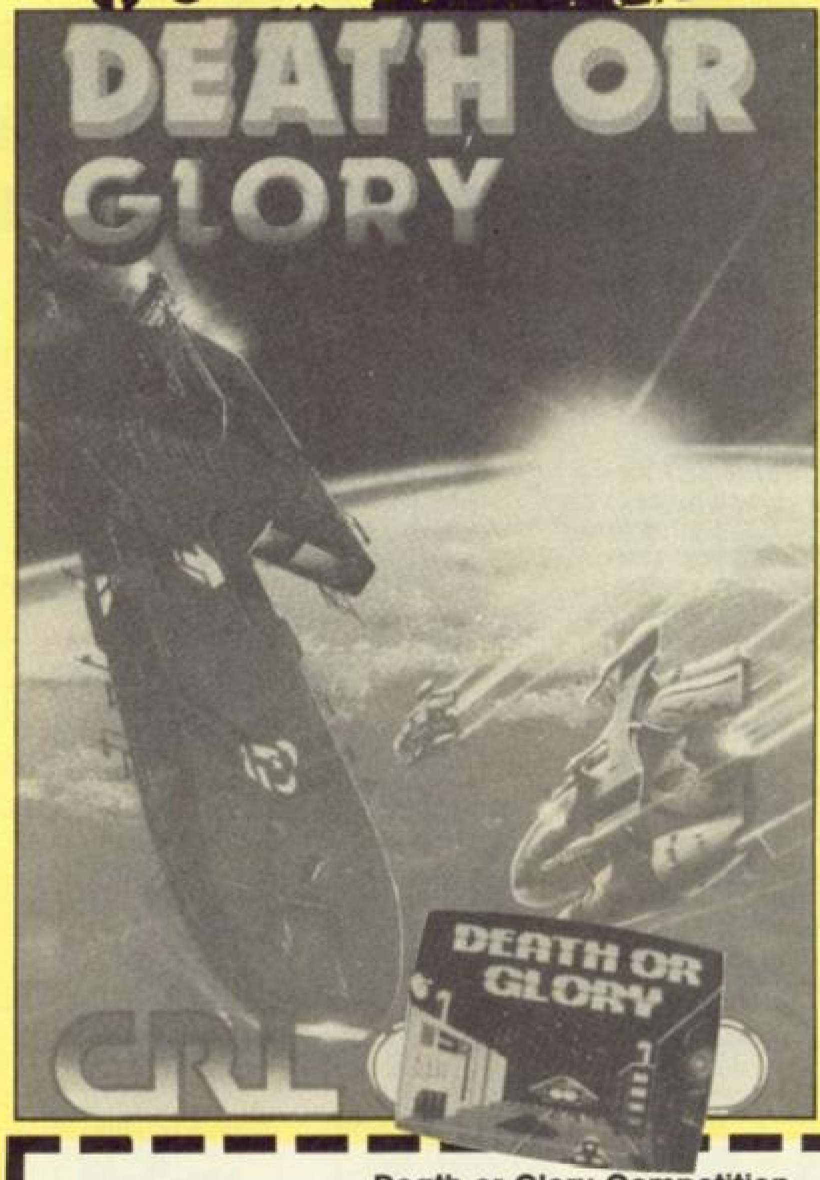
a) Mars
b) Earth
c) Saturn

3) What is the speed of light?

a) 186,000 miles per second
b) 1,000,000 miles an hour
c) 750,000 miles a minute

How to Enter

Write your answers on the coupon provided and send your entry to Death or Glory Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.
The competition is open to all readers of ZX except employees of Argus Specialist Publications, Chase Web and CRL. The editor's decision is final and no correspondence can be entered into. Please remember to write your answers on the back of your entry envelope. The closing date is Friday March 6th.



Death or Glory Competition

The answers to the cosmic questions are,

1) ....

2) ....

3) ....

Name: ....

Address: ....

....

....

Send your entry to Death or Glory Competition, ZX Computing, No 1 Golden Square, London W1R 3AB.

ACROSS THE POND

Mark Fendrick looks back at '86

Where have we been and where are we going as far as the Sinclair community in America is concerned? As we start a new year, it appears as if 1987 will be a very trying time.

Last year started out with hope on the horizon as the newest member of the Sinclair product line was becoming readily available. Although the QL had been around for a number of months, it previously had only been handled by Sinclair USA and American Express, and unless you had an American Express card, Sinclair was the only supplier. By January however, the established Sinclair dealers were now being permitted to stock the QL and related hardware and software.

For a while, it looked as if some life was going to be breathed into the American Sinclair marketplace. As dealers started to sell the QL, we saw Sinclair once again personally involved in North America distributing an actively produced computer. We had hoped that this was what we had been waiting for and that Sinclair would now take its place amongst the recognised computers in the United States. We had high hopes for the QL which had originally been priced at $499.00 but was now selling for $299.00. A matching printer and RGB monitor were also available (bearing the Sinclair QL logo) and a package containing all three — as well as the Psion suite of programs — cost only $795.00. Quite a bargain for a lot of computer.

Sinclair Show

In May, the second Sinclair related computer show ever was held in Cincinnati, Ohio. It originally started out as a proposed gathering of Sinclair owners in the midwest — but soon grew to proportions never envisioned by its organizers. Dealers from all parts of this country and Canada rented space and the original display area had to be doubled. What originally was supposed to be a local get-together attracted visitors from every section of both countries as well. For two days in May you would have thought that Sinclair computers were as popular here as they were in the United Kingdom.

If there was any doubt that even the ZX-81/TS 1000/TS computers were still in use by the faithful, they were put to rest during the exhibition. We knew that the T/S 2068 was still in use, but the interest in the ZX-81 computers surprised us all.

Once again, however, the QL found itself at the forefront of interest. Just a few weeks prior to the show two announcements changed the direction of the marketplace as we knew it; the sale of Sinclair's computer business to Amstrad and the purchase of the entire Sinclair North American stock by A+ Computer Response. Although no word had been officially given by Amstrad, it was (and still is) generally believed that they will not introduce current or future Sinclair computers into the North American marketplace. Amstrad itself will not even answer questions about its future in America.

But in the afterglow of this incredibly uplifting weekend, good things were once again predicted for the Sinclair marketplace in North America. A+ Computer Response was going to set up a network of authorized dealers and actually advertise the QL. For a while it looked as if we were finally going to come into our own. At one point there were seventeen



authorized dealers and print advertising started to appear. But, after a few months the advertising started to disappear and displeasure has started to be heard from the authorized dealers. A+ has started to offer merchandise direct to the public in conflict with the original understandings with the authorized dealers, as well as introducing QL kits for well under the price of a completely assembled QL (The kits do not come with either the Psion suite or a user's manual). Although the authorized dealers carry the kits as well, the price of a full QL — assembled and with the software and users' manual — is now as low as $209.00.

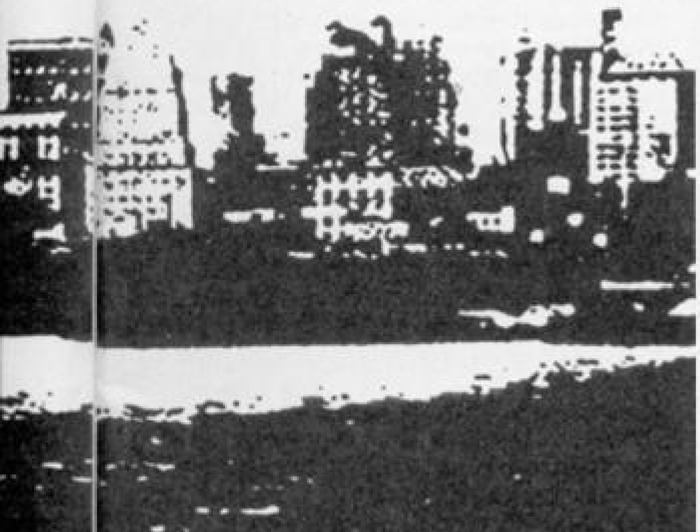
New products are starting to appear for the QL, a few of which have been developed here in the States, but the majority of which must still be imported from the UK. The first American piece of QL hardware has made its appearance in the shape of the QL Talker.

QL Talker

The Ql Talker, as the name implies, is a speech synthesizer. Unlike some earlier devices for the ZX-81 and T/S 2068 computers, this one requires just the device itself — no additional hardware (such as an amplifier and speaker) or software. They are all built into the device in the form in which it is purchased. All you have to do to set up the Talker is connect it to the serial port, open a channel to that port and print to it. It's that simple. Once that is done, the QL Talker then reads the string(s) sent to it, goes to its built-in dictionary for the proper sounds and produces the right words (nearly all of the time). Although there are occasions when you will have to spell some words phonetically, the QL Talker produces the proper sounds more often than any synthesizer software that I have previously encountered.

There are many uses for the Talker, and some of the more unusual ones are quite inventive. I have seen one routine for ARCHIVE which makes use of the Talker for interactive sessions. During a particularly long search, one company has programmed ARCHIVE to verbally call and inform when the search has been completed, eliminating the need for the businessman to sit and stare at a blank screen while the search is being completed.

Although a bit strange sounding, by copying the QUILL file to SER2 (with the QL Talker connected) I have heard this column being read to me. Any file can be read in this manner, although it should be text only to avoid control codes and the like.



Software and hardware for the T/S 2068 (the American version of the ZX Spectrum) is still being produced as well, and 1986 saw its share in this area as well. A look at some hardware comes first. When the ZX-81 was introduced, one of the main criticisms was its small amount of memory. The onboard 1K which came with the ZX-81 was increased to 2K when Timex introduced the American version — the T/S 1000. This was later increased to 16K onboard in 1983 when the T/S 1500 was introduced. When personal computers first appeared on the scene, 16K RAM was considered massive. Even the 1K and 2K which the earlier computers carried was an achievement. Consider the fact that even an unexpanded ZX-81 is more powerful than EMIAC — the first computer which caused city lights to dim each time it was used! But even if you used a ZX-81 you were still able to expand your computer. The most popular expansions in those days came from Memotech, and for about $200.00 you could expand your $99.00 computer to a full 64K. And if that were not enough for you, a system was available with which you could further expand your ZX-81 to one megabyte! Of course this would cost over $1,000.00 and you could buy a true business computer for that price. But when the T/S 2068 was finally released, it came with 72k built in, an claimed the ability to be expanded to one megabyte without a lot of expensive hardware. In fact, all you would need would be the actual memory chips properly configurd as the bank switching capability was already resident in the computer. But no T/S 2068 memory expansion was planned, and until 1986 none were developed.

This T/S 2068 compatible memory expansion takes the form of a command cartridge and as such fits into the port on the front of the computer. One of the features of this unit which makes it popular is the lithium battery onboard. By continually supplying power to the RAM chips, this board offers non-volatile memory. Programs and/or data are immediately available upon power-up without the need to load from any outside source.

CP/M

Speaking of mass storage devices, as reported a few months ago, CP/M compatibility was introduced for the T/S 2068 during the year. Combining their long standing design in floppy disk interfaces for both the T/S 1000 and T/S 2068, AERCO has opened up the world of popular CP/M software for the T/S 2068 user. Once the most utilized operating system, there is a great deal of CP/M software available — both professional (such as Wordstar) and public domain. Any CP/M program in Morrow format will now be able to run on the T/S 2068.

Developments for the T/S 1000 have not completely disappeared either, and in 1986 a handful of products still made their way to market for these computers. Graphics, telecommunications and extended BASIC were the top attention getters in 1986. Even the T/S 1500 — which never had a chance to make its mark in the general scheme of things — had software developed specifically for it. This takes the form of a high resolution dungeon game by the name of Dungeon of Ymir. It contains 24K of machine code which creates nine levels, sixteen types of monsters, fourteen objects and six spells. Thanks to built-in routines you can save games in progress and load them back in seventy seconds. Various versions are available depending on what hardware setup you have.

There continued to be a large demand for Spectrum software due to the continued popularity of various Spectrum Emulators. However, while ZX-81 software runs fine on T/S 1000's and T/S 1500's, and Spectrum software is compatible with Spectrum emulated T/S 2068's, the QL — with its JSU ROM — still has to rely on modifications to British software before 100% compatibility is achieved.

Where will 1987 take us? It's hard to say. After all, many observers (myself included) have prematurely written the Sinclair computers' obituary in the past. While things are not as bright as they were in January, 1986, the North American Sinclair faithful have the tendency to confound the pundits. Hopefully 1987 will not prove the exception to what has happened in the past.

The Friendly Programmer

When someone else's program crashes on you it's "bad programming" but when your own program crashes who can you blame? User friendliness begins at home as Alan Davis explains. . .

Let's begin with a tall story, but one with a moral. It's called "The Shape of Things to Come?". . . .

Sid had just bought a book, and on arriving home he curled up in his favourite armchair to enjoy it. Unfortunately it was one of those new-fangled books fitted with a complicated anti-piracy device to defeat photo-copiers, and so it took him a quarter of an hour to get it open. Still, he did manage it in the end, and he settled down to read. After he'd read a couple of chapters, his fingers accidentally turned two pages over instead of one. Now the designer of the book hadn't anticipated that anyone could do such a silly thing — so the page-opening mechanism jammed, and the book snapped shut. "Oh well, my fault" said Sid, philosophically. (It took him another 14 minutes to get it open again.) This time he took great care not to turn two pages at a time, but after a while he just couldn't resist trying to sneak a look at the last page to see if it really was the butler who did it — and at that point all the pages fell out. . . . .

Listing 1

```
  1 REM ***SIMPLE MENU ROUTINE
  2 REM
  3 REM
 10 CLS : PRINT AT 8,11;"OPTION
S";AT 10,5;"1: Do something"'TAB
 5;"2: Do something else"'TAB 5;
"3: Do something different"
 20 PRINT #1;AT 0,8; INVERSE 1;
"PLEASE SELECT..."
 30 PAUSE 0: LET i$=INKEY$
 40 IF i$<"1" OR i$>"3" THEN  G
O TO 30
 50 BEEP .1,30
 60 GO SUB 100*VAL i$
 70 GO TO 10
100 CLS : PRINT AT 10,4;"I've d
one something"
110 GO TO 310
200 CLS : PRINT AT 10,2;"I've d
one something else"
210 GO TO 310
300 CLS : PRINT AT 10,2;"I've d
one something different"
310 GO SUB 500
320 PAUSE 0: RETURN
500 PRINT #1;AT 0,8; INVERSE 1;
"PRESS A KEY...": RETURN
```

Now I know that this is an odd way to begin an article on Spectrum programming. But substitute "software" for "book", and I think you'll see what I'm getting at. We've all bought (and perhaps — heaven forbid! — even written) programs which were poorly error-trapped, crashing without warning as the result of an injudicious key-press; we're all distressingly familiar with those barbaric "protection" methods which hang up the machine when you press BREAK; and of course the monstrous LENSLOK has found no difficulty at all in achieving a place in the top ten list of contributors to the misery of mankind.

Only one person suffers from all this: the user of the program. In principle, there's no reason why he shouldn't find using a computer program almost as straightforward as reading a book, but the fact is that many programs give him a rough ride. So I thought it might be a good idea if we tried to find a few ways of making life easier for him when writing our own programs. The keyword, then, is "friendly", and friendliness is always worth striving for, no matter what kind of program you're writing. It doesn't even matter if the program is a utility being written only for your own use — because we all make mistakes; and the last thing you want is a poorly crash-proofed utility which leaves you in a mess after maybe hours of work, just because you pressed the wrong key by accident.

Weak links

Usually, the friendliness of a program isn't determined by ingenious programming. Rather, it depends on the programmer painstakingly searching for possible weak points and anticipating potential errors in such a way that they're rendered harmless to the program — and by implication, harmless to the user. From our present point of view, the danger points will occur at places in a program where some kind of input is needed from the user and it's predominantly this area that we'll be looking at in this article.

On the whole, the friendliest way of getting input from the user is probably to present him with a menu of options, and ask him for a single prod at the keyboard to make his selection.

This automatically puts a limit to the silly things he might try to do, and has the great advantage of being easy to understand. It also means that our task of error-trapping is made very straightforward. **Listing 1** is the sort of routine one might use here. It presents a choice of three actions (pointless ones, here — but this is only an example) determined by pressing key 1, 2, or 3 — and is about as simple to operate as any program could be. In fact, short of pressing BREAK, the user simply can't crash the program — because line 40 rejects every keypress except the three allowable ones.

Wherever it's appropriate a menu-driven utility program gets my vote every time. But programming (like life, alas) often presents us with circumstances which can't be tackled in quite the way we might like. Sometimes a single key-press menu just won't do, and this is generally the point at which our program's "friendliness" can start to acquire rough edges. I can't cover all eventualities, of course — it'd take a lifetime! But we can

Listing 2

```
  1 REM ***SIMPLE INPUT ROUTINE
  2 REM
  3 REM
 10 INPUT "How many would you l
ike?"'x
 20 IF x>50 THEN  PRINT "Sorry,
 that's too many.": GO TO 10
 30 PRINT "OK. You have ";x;" o
bject";"s" AND x<>1
 40 GO TO 10
```

Listing 3

```
   1 REM ***ERROR-TRAPPED INPUT
   2 REM
   3 REM
  10 INPUT "How many would you l
ike?"' LINE z$: GO SUB 500
  20 IF fail THEN  PRINT "Number
s only, please.": GO TO 10
  30 IF x>50 THEN  PRINT "Sorry,
 that's too many.": GO TO 10
  40 PRINT "OK. You have ";x;" o
bject";"s" AND x<>1
  50 GO TO 10
 500 LET fail=0: IF z$="" THEN
LET fail=1: RETURN
 510 FOR i=1 TO LEN z$
 520 IF z$(i)<"0" OR z$(i)>"9" T
HEN  LET fail=1: RETURN
 530 NEXT i
 540 LET x=VAL z$
 550 RETURN
```

learn a good deal by taking one specific example and delving into it thoroughly — because it's really the thinking process underlying this which is important, rather than the example itself.

The example I've chosen is one which commonly arises in programs of many types, namely, where the program requires a number (which may be several digits long) to be entered by the user. We'll restrict the discussion to integers, here — and let's also add the arbitrary condition that for some reason peculiar to the situation the number mustn't exceed 50. This is just the kind of thing which could arise in a typical programming situation.

On the face of it, **Listing 2** would appear to be a straightforward answer to the problem. It uses the simple BASIC "INPUT" command to assign a value to "x", checks to see whether the value entered is permissible (x mustn't exceed 50, remember), and prints an appropriate comment. It works, of course; but as a piece of "friendly programming" it's a dead loss because there are so many possible ways of crashing it. Just for starters, try entering a letter, or several letters, or even (since there's nothing to stop you) something like VAL z$! Gruesome, isn't it?

Improving input

What can we do to improve matters? Well, a solution which sorts out the major difficulties is given in **Listing 3**. We're still using the INPUT command — but now we're picking up the entry not as a number, x, but as a string, z$. This gives more power to our elbow, because we can now add a little error checking subroutine (lines 500-550). This rejects the input if the entry is either an empty string, or if any of the characters are not pure numbers between 0 and 9, and returns to the user with an appropriate comment and a repeat request. On the other hand, if all is well, line 540 assigns the correct value to x, and Bob's your uncle.

This isn't a bad solution to our problem, in fact — and you may well consider it good enough. But it's by no means perfect, because the INPUT command still has a couple of nasty tricks up its sleeve. Try pressing CAPS SHIFT/6, for example.

Alternatively, type in lots of numbers — say a couple of rows. Oops! So unless you have a quaint fondness for the "STOP in INPUT" and "Number too big" error reports, it looks as though we'll have to continue our search for the ultimate in friendly input routines.

Obviously, to improve matters further, we'll have to abandon the INPUT command altogether, and simulate a similar command of our own which allows the program to intercept every character as it's typed. This could be tackled in several ways, and **Listing 4** is one of them. The main bulk of the error-trapping is done in line 20; because we're reading the keyboard using INKEY$, we can examine each keypress as it comes, and ignore it if it isn't either a number (CHR$ 48 — CHR$ 57), ENTER (CHR$ 13), or DELETE (CHR$ 12). This still leaves quite a lot to be done, though. The variable "i" keeps count of the number of valid characters typed, so that line 30 can prevent the user attempting to ENTER or DELETE a non-existent number. Line 70 solves the problem of the "Number too big" error by limiting the number of digits that can be typed to 5. Line 80 builds up the string z$ one character at a time, and when ENTER is pressed, line 90 extracts the value of x (i.e. VAL z$) that we need. (Line 500, by the way, is the DELETE subroutine.)

Although there are alternative lines of approach, this seems to be about as far as you could go in BASIC. It's as crash-proof as the menu routine in Listing 1, in that only BREAK will defeat it. If you type it in and try it, though, you may notice a marginal sluggishness in the keyboard response. It's only very slight, but should you wish to add extra checks between each keypress (for some other specific application) it could become irritating. The problem arises of course because the more work you ask the program to do between each keypress, the longer it will take to do it.

The only comprehensive solution to this is to read the keyboard and do all the "between keypresses" error checking in machine code, returning to BASIC only when the input is ENTERed. One way of tackling this is the assembler program I've given in **Listing 5**. This works by reading the system variable LAST-K, which stores the code of the last newly pressed key. If this is non-zero, indicating a keypress, then the error-trapping checks are called one after another so that only a valid keypress is accepted. The routine also looks after the printing of digits to the screen, and the deleting process. The actual input is stored in a series of up to 5 bytes starting at address STORE.

A simple machine code program like this will need a short BASIC subroutine to drive it — such as the one I've given in **Listing 6**. If you want to try this out for yourself you'll need the machine code residing in memory at 65000 — don't forget to CLEAR 64999 beforehand. This new BASIC/machine code combination will behave in every way like Listing 4, except that all trace of keyboard sluggishness has disappeared, together with the added bonus that even BREAK is disabled during the period where an input is being requested. There is absolutely nothing at all that the user can do which will cause a crash. Error-trapping is complete, and the only improvement in the way of friendliness would be to include more detailed prompt messages on screen.

As I said earlier, it's the general process involved which is important, rather than the details; and I hope that the method of progressively isolating the problems and then solving them is clear from the examples I've used here. The extent to which you go will largely depend on how likely it is that

Listing 4

```
   1 REM ***CUSTOM INPUT ROUTINE
   2 REM
   3 REM
  10 LET z$="": LET i=0: PRINT #
1;AT 0,0;"How many would you lik
e?"
  20 PRINT #1;AT 1,i;">": PAUSE
0: LET i$=INKEY$: LET k=CODE i$:
 IF (k<48 OR k>57) AND k<>13 AND
 k<>12 THEN  GO TO 20
  30 IF NOT i AND (k=13 OR k=12)
 THEN  GO TO 20
  40 BEEP .05,30
  50 IF k=12 THEN  GO SUB 500: G
O TO 20
  60 IF k=13 THEN  PRINT #1;AT 0
,0,,,,: GO TO 80
  70 IF i=5 THEN  GO TO 20
  80 PRINT #1;AT 1,i;i$;: LET z$
=z$+i$: LET i=i+1: GO TO 20
  90 LET x=VAL z$: IF x>50 THEN
 PRINT "Sorry, that's too many!"
: GO TO 10
 100 PRINT "OK. You have ";x;" o
bject";"s" AND x<>1
 110 GO TO 10
 500 LET i=i-1: LET z$=z$( TO LE
N z$-1): PRINT #1;AT 1,i;"  ": R
ETURN
```

Listing 5

```
   1 REM ***CUSTOM INPUT ROUTINE
         (MACHINE CODE VERSION)
   2 REM
   3 REM
  10 LET point=65138: LET store=
65139
  20 PRINT #1;AT 0,0;"How many w
ould you like?"
  30 LET m=USR 65000
  40 PRINT #1;AT 0,0,,,,
  50 LET z$="": FOR i=store TO s
tore-1+PEEK point: LET z$=z$+CHR
$ PEEK i: NEXT i: LET x=VAL z$
  60 IF x>50 THEN  PRINT "Sorry,
 that's too many!": GO TO 20
  70 PRINT "OK. You have ";x;" o
bject";"s" AND x<>1
  80 GO TO 20
```

others will use your program, of course — and it will also depend on how familiar with computers they're likely to be! People can do very odd things when they're desperate. . . . and it's up to the programmer to safeguard the inexperienced user from himself.

Protection

I can't leave the subject of friendliness without saying a word about "protection". Some people do seem to get obsessive about this sometimes, and don't regard a program as finished until they've incorporated all the BREAK disabling tricks they can muster. If the result simply means that BREAK is ignored, then fine; this in itself protects the user from stopping the program and getting into a mess. But if pressing BREAK causes the machine to hang up, or a system reset, then surely this is the very opposite of friendly programming! Furthermore, such methods are actually quite pointless. They certainly won't make your program uncopyable, and if all you're worried about is to keep prying eyes from your code, then it stands to reason that anyone who's capable of understanding your program in the first place will have no difficulty in making short work of your protection scheme! The moral? Think twice before you plant a bomb in your program . . .

I'm aware of course that writing an article like this is rather like putting a gun to my own head, because it's almost certain that someone will now closely examine all my programming examples in previous articles to see if I take my own advice. But I'm happy to take the risk. A bit of egg on my face won't do any harm, and if the result is an increase in the total amount of programming friendliness in the world, then I guess it's worth it.

Letters on the subject should be sent to the editor — preferably with BREAK disabled and fitted with LENSLOK.

Listing 6

```
*HISOFT GENS3M2 ASSEMBLER*
       ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: 00

                 10 *D+
         20 *C-
         30 ; Assembler program to input up to 5 digit number
         40
         50
65000    60         ORG  65000
65000    70         XOR  A
65001    80         LD   (POINT),A; Reset the store pointer
65004    90         CALL #1601; Open channel to lower screen
65007   100 START   CALL PROMPT
65010   110         XOR  A
65011   120         LD   (LAST_K),A;Reset system variable LAST_K
65014   130 INPUT   LD   A,(LAST_K)
65017   140         CP   0
65019   150         JP   Z,INPUT; Wait for a key-press
65022   160         CP   12
65024   170         JP   Z,DELETE
65027   180         CP   13
65029   190         JP   Z,ENTER
65032   200         CP   58
65034   210         JP   NC,START
65037   220         CP   48
65039   230         JP   C,START; Only numbers get past here
65042   240         LD   (CHR),A; Temporarily store character
65045   250         LD   A,(POINT)
65048   260         CP   5
65050   270         JP   NC,START; Abort if more than 5 digits
65053   280         LD   HL,STORE
65056   290         LD   DE,(POINT)
65060   300         LD   D,0
65062   310         ADD  HL,DE; HL now points to correct address
65063   320         LD   A,(CHR)
65066   330         LD   (HL),A; Put character code into store
65067   340         CALL PRINT
65070   350         LD   A,E
65071   360         INC  A
65072   370         LD   (POINT),A; New pointer position
65075   380         JP   START; Back for another key-press
        390
65078   400 PROMPT  CALL SETPOS
65081   410         LD   A,62
65083   420         RST  16; Print a "prompt" symbol
65084   430         RET
        440
65085   450 PRINT   CALL SETPOS
65088   460         LD   A,(CHR)
65091   470         RST  16; Print current character
65092   480         RET
        490
65093   500 SETPOS  LD   A,22
65095   510         RST  16
65096   520         LD   A,1
65098   530         RST  16
65099   540         LD   A,(POINT)
65102   550         RST  16; Same as PRINT #0; AT 1,(POINT);
65103   560         RET
        570
65104   580 DELETE  LD   A,(POINT)
65107   590         CP   0
65109   600         JP   Z,START; Abort if nothing typed yet
65112   610         CALL SETPOS
65115   620         LD   A,32
65117   630         RST  16; Print a space over the old prompt
65118   640         LD   A,(POINT)
65121   650         DEC  A
65122   660         LD   (POINT),A; Move pointer back one notch
65125   670         JP   START; Back for another key-press
        680
65128   690 ENTER   LD   A,(POINT)
65131   700         CP   0
65133   710         JP   Z,START; Abort if nothing typed yet
65136   720         RET  ;Back to BASIC
65137   730 CHR     DEFB 0; Temporary store for character code
65138   740 POINT   DEFB 0; Stores current pointer position
65139   750 STORE   DEFS 5; Reserves 5 bytes to store digits
23560   760 LAST_K  EQU  23560

Pass 2 errors: 00

Table used:    147  from   300
```

HISOFT BASIC COMPILER

Hisoft Basic Compiler
HiSoft
£15.95

Hisoft are a company famous for their Devpac assembler which is the assembler all others are measured by. They are not a prolific producer of software, but usually when they do market something it is of a very high standard.

Now BASIC, as we all know, is useful for learning to program and is tolerable in some applications where speed is not essential. To get an arcade type of program to run at a reasonable speed on the Spectrum you need to program in machine code and that takes a fair bit of serious study.

Alternatives are to use a special language such as White Lightning or to get a compiler to change BASIC into machine code.

The latter has been attempted with reasonable success by PSS with MCODER 1, 2 & 3, the last being an excellent compiler with few disadvantages, the biggest being the compiled code is not relocatable and it is not possible to combine several routines compiled separately. BLAST on the other hand was a disaster and its name aptly mirrored its purchaser's expletives.

Floating point

The HiSoft compiler is a full floating point (copes with decimals as well as integers, ie, whole numbers), compiler for **all** the Spectrum variants. The code itself occupies around 11K.

HiSoft claim this means that programs up to 30K can be compiled in one operation, 128 and PLUS 2 owners have a slightly modified operating system which takes advantage of some of the capabilities of the machines.

The compiler code is located low in memory between the system variables and the microdrive map area. Memory maps of the program's requirements are given in the manual.

Limitations are few and are unlikely to be restrictive unless you require them for a specific programming purpose. These are that no expressions are allowed in DATA statements, evaluations of string variables (eg. VAL x$) is not allowed and arrays of three or more dimensions cannot be used.

There are also a few system commands which are not allowed but these can be overcome due to the compiler's ability to move in and out of BASIC selectively, these are such commands as LOAD, SAVE, NEW etc.

Although supplied on tape there are instructions for making a Microdrive or disk backup copy, a considerate and useful option.

The compiler has sixteen directives although often only one or two may be required and these are used by adding REM: lines before the start of the code to be compiled.

One compilation is initiated then all relevent info is provided, including entry points if the program was split into sections, and at the end the start and length of code for saving to tape etc is given.

On Test

First the 50 page manual deserves a congratulatory mention. I took it away and read it and due to its step by step examples I understood it without any problem and felt confident when I went back to the computer.

My first difficulty came when I loaded the program and tried to convert it to run on my disk drive system, it wouldn't! Then I tried running it direct from tape, it locked up!

Only when I disconnected my drive would it work, now the problem is that I am using a TRL Beta interface and it is obviously incompatible with it, I assume that the Disk drive they refer to in the manual is the Opus Discovery which is compatible with the microdrive system. A bit of a disappointment.

The program was now working and the short demos/examples worked perfectly and impressively. This is the only compiler I know which can handle Sinclair's computed GOTO and GOSUB features.

Speed limits

The speed increase was variable but this is explained, and reasonably so, as being dependent on the number of 'real' number calculations needed and use of the PLOT/DRAW functions. By specifying integer and positive integer variables at the start by using the directive REM: INT variable, list the speed increase is optimised.

Error messages are clear, detailed in the manual and are produced on either of the first two of the three passes the program makes.

My next test was to try and find a BASIC program on tape (all mine being kept on the inoperable disk drive) eventually I found a couple and of course the first included a DIM A(2,4,3) instruction which is not allowed.

The next was a copy of an earlier ZXC listing called Platform Jack, a rather slow jump game. It compiled after three attempts when I had to make minor amendments to RUN statements and gave a speed increase of about 20 times! Completely unplayable without the addition of some delay loops.

Out of the four other games I experienced no problems and all gave an impressive gain in speed. I did find that in general the code produced tended to equal or become slightly longer than the original and this meant that in practical terms around 11K to 15K was compilable in one go. For really long programs then it may be possible to break them into smaller units and compile to specific addresses. A special facility to compile DATA and program separately is built into the program.

I am proud to note that ZXC is credited at the end of the manual for allowing the use of one of Toni Baker's published routines for the 128 keypad simulator.

Probably the most versatile general purpose compiler on the market it has a little more flexibility than PSS' Mcoder 3 which must be its nearest rival.

It could certainly produce commercial quality programs of some types, but to produce that state-of-the-art graphic arcade masterpiece is unlikely without custom written machine code routines.

If you're serious about your computing then this is the program you've been waiting for, as close as anyone is likely to get to an easy to use, most features supported compiler for the Spectrum computers.

THE DISCOVERY COLUMN



John Wase presents another selection of useful routines for Discovery owners.

The first routine this month is a neat little COPY routine from John Bennett, who lives near Bedford, that enables you to dump a screen to a printer. Funny, I was asking for this sort of thing only in last month's column, and before it's even printed, here it is! The program **(Figure 1)** is short, easily typed in, *and it works!*. Line 30 connects stream 3 to the printer driver software in the Opus port. Line 100 switches the printer spacing to 8/72" (you might need to adjust this for your printer) and 120 switches the printer to graphics mode. Line 140 does the plotting for one pass of the print head and line 160 a line feed. The spacing is returned to normal in line 180 and line 190 closes the stream. And there it is, complete with a SAVE routine at the end. **Figure 2** is a picture from my old tape of Lunar Jetman produced by this routine. Although it was originally written for a Shinwa CP80, it works fine on my Epson FX80, too. The only problem is, as John says, that it's rather slow: some machine code might come in useful. How about it, folks?

Morefiles

Steven Nutting of Histon, of "supercat" fame has sent in a program which took my eye. This is another program for playing with the CATalogue file, somewhat more explicit than the very short BASIC routine which I published a month or two ago, and it includes a bit of machine code to speed things up a litte.

There are 719 sectors on each disc numbered 0 to 718, and each sector can hold 256 bytes of CAT or program information. The CAT file uses the first seven sectors (numbered 0 to 6). The remainder of the disc available for use is laid out as follows: the disc formatting information and disc name use the first 32 bytes, and each subsequent file uses 16 bytes. This means that there is room for information on 110 programs, data files, pieces of code, etc; (32 + 110 * 16 = 1792 bytes = 1.75K = 7 sectors). The fourth byte in the first of those CAT sectors contains a value equal to the number of the last sector used by the CAT file (normally extending from sector 0 through to the end of sector 6: so the normal value is 6, indicating 7 sectors). However, by inserting another value in this 4th byte, we can extend the number of sectors in the CAT file up to a maximum of 43. This allows us to store a miximum of 676 files in the CAT file in addition to the first 32 bytes which contain the disc name and formatting information. This, in turn, leaves us 676 sectors. As it is not possible to store more than one program file or what-have-you on a sector, the maximum number of programs, etc, which can be stored is 67, just equal to the maximum number of files in the CAT file, and probably enough for most people.

The program

To alter the magic byte, the program listed in **Figure 3** first of all needs to wipe the disc clean with a FORMAT command. So *don't* use this program on a disc which contains valuable routines. Next, the program loads in sector 0, alters this byte, and saves this sector back to where it belongs.

Lines 10 to 30 set up the 40 byte machine code routine, protected by CLEAR 65069 in line 10. Lines 40 to 60 ask you to INPUT the drive number, check that this is reasonable, and then POKEs it into the machine code routine. Lines 70 to 80 ask for the disc name; f$ contains seven letters, the maximum number allowed by this routine. Next, in lines 90 to 100, you are asked to INPUT the number of files, f, which you need to save on the disc: this number is also checked to see if it is reasonable. Line 110 works out the approximate number of CAT sectors needed for storing the number of files indicated by the value contained in f; this is then POKEd into the machine code routine. Then, in line 120, the disc is FORMATted with the name, f$, plus the number of files which can be saved. RANDOMIZE USR 65080 calls the machine code routine **(Figure 4)**, and, to finish off, a CAT confirms the filename, the number of files available, and the memory left for the programmer.

Code

The machine code in Figure 4 has comments alongside each step and is pretty self-explanatory. It loads the selected sector from the disc into the Spectrum's RAM, starting at address 65110 (see assembly listing, line 14), having first loaded HL with the number of the CAT sector required; in this case, the first sector, number zero (line 12). The number of CAT sectors required is 'poked' into address 65114 (line 3) and the sector is saved back to the disc: finally (last line) the Spectrum ROM is paged back in with a return to BASIC.

Fastload

This program, by Neil Hewitt of Coventry is one of a whole clutch of loader-programs and/or catalogue programs which I have received lately, several of which you've already seen. I therefore have to be rather selective. Although it has some limitations, Neil's program stood out because of the impressive presentation; the 128 style windows, in particular, are most attractive.

Fastlist

The program in **Figure 5** is all in BASIC. Type it in, using graphics A in place of the question marks in line 9993 and save it as "run", auto running from line 1. It

allows you to LOAD programs, ERASE files, gives you an extended CAT, COPY files, and so on. The menu bar will (see **Figures 6 and 7**), appear at the top of the screen with the first option (LOAD), flashing. The cursor keys move the choice along the menu bar highlighting each option in turn. The required option is selected by ENTER, ZERO or DELETE. A pull-down type menu then appears, selections being made just as on the 128 menus. To whet your appetite, Figures 6 and 7 show screen dumps of two of the menus, the LOAD menu (blue background) and the MISCELLANEOUS menu (green background). You leave a menu by pressing Q to quit — the only prompt not on-screen, so take heed.

The options are straightforward — you can load, or load and run programs or code, or erase any file, and there is a miscellaneous section which includes various resets. the code-length of the files. Files beginning with CHR$ 0 to hide them in the catalogue printout show up here with a question mark at the beginning of the filename: a LOAD and ERASE option is provided to deal with these, too (in these options, only the name, without the question mark, should be entered).

The COPY section is a bit limited, using as it does the rather limited MOVE command; so it will only COPY individual programs and not the whole disc between discs of two different sizes. Those who might run up against this problem should use a program like Jose Pedro's "opuscat". In spite of this (and it's easy for me to be critical) this program is really

Figure 1

```
   1 REM OPUSCOPY: JOHN BENNETT
   5 BORDER 1: PAPER 6: CLS
  10 REM OPUS COPY/SHINWA CP80
  20 PRINT AT 5,8;"OPUS PRINTER
COPY ROUTINE 1986": PRINT ''
  30 OPEN #3;"b"
  35 PRINT                       "
Set your printer up first then
LOAD your picture or SCREEN$.
The printer will start up
automatically after 15 seconds.
It will then take about six
minutes to print a full screen"
  40 INPUT "ENTER TITLE OF SCREE
N TO LOAD  ";N$
  45 PRINT #1;AT 1,0;"  Load fro
m Tape or Disc? (T/D) ": PAUSE 0
  50 IF INKEY$="d" OR INKEY$="D"
 THEN GO TO 70
  55 IF INKEY$<>"t" AND INKEY$<>
"T" AND INKEY$<>"d" AND INKEY$<>
"D" THEN GO TO 45
  60 LOAD N$SCREEN$ : GO TO 100
  70 LOAD *1;N$SCREEN$
  90 REM Copy
 100 LPRINT CHR$ 27;"A";CHR$ 8;
 110 FOR H=175 TO 7 STEP -8
 120 LPRINT CHR$ 27;"K";CHR$ 0;C
HR$ 1;
 130 FOR W=0 TO 255
 140 LPRINT CHR$ (128*POINT (W,H
)+64*POINT (W,H-1)+32*POINT (W,H
-2)+16*POINT (W,H-3)+8*POINT (W,
H-4)+4*POINT (W,H-5)+2*POINT (W,
H-6)+POINT (W,H-7));
 150 NEXT W
 160 LPRINT CHR$ 10
 170 NEXT H
 180 LPRINT CHR$ 2
 190 CLOSE #3
 200 STOP
9996 STOP
9997 SAVE "opuscopy" LINE 1
9998 STOP
9999 SAVE *1;"opuscopy" LINE 1
```

Figure 2



Figure 3

```
   5 REM
     ***M O R E F I L E S***
        BY STEVEN NUTTING

  10 CLEAR 65069: LET c=0: FOR a
=65070 TO 65109: READ n: POKE a,
n: LET c=c+n: NEXT a
  20 IF c<>3257 THEN PRINT "Data
 Error"
  30 DATA 6,2,205,58,254,62,42,5
0,90,254,6,0,197,205,8,23,6,0,24
7,18,34,81,254,330,0,17,86,254,1
4,0,193,62,1,205,0,0,195,72,23
  40 INPUT "Drive (1 or 2): ";d
  50 IF d<1 OR d>2 THEN GO TO 40
  60 POKE 65103,d
  70 INPUT "Filename (max 7 lett
ers): "; LINE f$
  80 IF LEN f$<1 OR LEN f$>7 THE
N GO TO 70
  90 INPUT "Number of files (1-6
76) :";f
 100 IF f<1 OR f>676 THEN GO TO
90
 110 LET s=(f+2)*16/256: POKE 65
076,INT s
 120 FORMAT d;f$+STR$ f: RANDOMI
ZE USR 65070: CAT d
```

The catalogue section is quite versatile: it is printed in three columns so that it doesn't easily go off-screen (we had two last month, but I guess three is the limit with Sinclair's Rom lettering) and there is an option to print

Figure 4

```
DISSAMBLY of addresses  65070-65109 (40 bytes)

                        ORG 65070
65070 006,002             LD B,2          ;Used by Discovery Rom to say load in sector.
65072 205,58,254          CALL loadsave   ;Load or Save sector depending on value of the B reg.
65075 062,000             LD A,0          ;Poke 65114 with the number of Cat sectors used to hold file data (no in A reg)
65077 050,090,254         LD (65114),A
65080 006,000             LD B,0          ;Used by Discovery Rom to say save sector.
65082 197       loadsave: PUSH BC         ;Save B reg for later use
65083 205,008,023         CALL 5896       ;Call Discovery Rom in.
65086 006,000             LD B,0          ;Find from Discoverys Lookup Tables   the I/O Read & Write routine.
65088 247                 RST #30         ;Then Load HL with the address.
65089 018                 DEFB 18
65090 034,081,254         LD (add),HL     ;Poke the Address in the loadsect routine.
65093 033,000,000         LD HL,0         ;LD HL,sector no 0 (The start of the CAT sectors).
65096 017,086,254         LD DE,65110     ;Load sector at address 65110.
65099 014,000             LD C,0          ;Load in 256 bytes.
65101 193                 POP BC          ;Retrieve the B reg that was saved,if B=2 then load sector of if B=0 then save.
65102 062,001             LD A,1          ;Select drive 1.
65104 205                 DEFB 205        ;This would turn out to be CALL nn
65105 000,000             DEFW 0
65107 195,072,023         JP 5960         ;Block SINCLAIRS ROM back in and RET to Basic.
```

very neat, and for a mere BASIC job packs in a lot of features and presents them very nicely. Try it!

Rename

This is a neat little routine from Tom Nicholson of Motherwell which will rename either files stored on disc or even the disc itself. It is written for a single drive disc: for drive 2 you merely have to alter line 30. The listing **Figure 8** is short and to the point. When you type in line 130, capital N, capital S and capital R should all be in inverse video, this gives a very neat and professional presentation.

Finally, please can I appeal to you to submit programs *on tape or preferably on disc*. How about these code dump programs, then, and some larger size dumps, too, with pretty shading? And don't forget the random access facility. Several of our readers have had difficulties with filing programs, so how about a good, well documented one of your own? Keep them coming in. See you next month.

Figure 5

```
   1 DIM q$(1,32)
   2 DIM w$(5,2): DIM d$(5,2): D
IM p$(5,1): DIM i$(5,1): DIM l$(
5,32): DIM c$(5,19,30): DIM y$(5
,2): DIM x$(5,2)
   3 LET leng=0: LET factor=1
   4 LET b$="run": REM BOOT NAME
   5 DIM d$(5,2): DIM w$(5,2)
   7 LET drive=1
   8 FOR f=0 TO 7: READ a: POKE
USR "a"+f,a: NEXT f: DATA 255,25
4,252,248,240,224,192,128
  10 DIM i$(5,1): DIM p$(5,1): D
IM x$(5,2): DIM y$(5,2): DIM l$(
5,30): DIM c$(5,10,30)
  20 FOR f=1 TO 5: READ p$(f),i$
(f),w$(f),d$(f),x$(f),y$(f),l$(f
): FOR g=1 TO (VAL d$(f))-3: REA
D c$(f,g): NEXT g: NEXT f
  30 DATA "5","0","18","7","2","
1","Load Menu","Load Program","L
oad Code","Load Code & Run","Loa
d Hidden File"
  31 DATA "4","0","19","5","2","
4","Erase Menu","Erase File","Er
ase Hidden File"
  32 DATA "5","0","17","7","2","
12","Cat Menu","Cat Drive 1","Ca
t Drive 2","Cat 1 & Lengths","Ca
t 2 & Lengths"
  33 DATA "4","0","18","7","2","
12","Copy Menu","Copy Drive 1 to
 2","Copy Drive 1 to 1","Copy Di
sc 1 to 2","Copy Disc 1 to 1"
  34 DATA "4","0","15","8","2","
15","Misc","Reset System","Reset
 Drives","Set IF1 Map","Boot Pro
gram","Change Drive"
  40 PRINT AT 0,0; INVERSE 1;"Lo
ad | Erase | Cat | Copy | Misc"
  49 LET opt=1
  50 PRINT AT 0,0; INVERSE 1;"Lo
ad | Erase | Cat | Copy | Misc":
 RESTORE 60: FOR f=1 TO opt: REA
D y,o$: NEXT f: PRINT AT 0,y; FL
ASH 1;o$
  51 IF INKEY$="" THEN GO TO 51
  52 IF INKEY$=CHR$ 9 AND opt<5
THEN LET opt=opt+1
  53 IF INKEY$=CHR$ 8 AND opt>1
THEN LET opt=opt-1
  54 IF INKEY$="0" OR INKEY$=CHR
$ 13 OR INKEY$=CHR$ 12 THEN GO T
O 70
  55 GO TO 50
  60 DATA 0,"Load ",6," Erase ",
14," Cat ",20," Copy ",27," Misc
"
  70 LET wn=opt: GO SUB 9990: GO
 SUB 8000
  80 LET leng=0: LET factor=1: G
O SUB wn*1000: FOR f=2 TO 20: PR
INT AT f,0;q$(1,1 TO ): NEXT f:
GO TO 50
1000 GO TO 1000+option*10
1010 INPUT "Program Name:"; LINE
 n$: LOAD *drive;n$
1020 INPUT "Name:"; LINE n$: LOA
D *drive;n$CODE
1030 INPUT "Name:"; LINE n$: INP
UT "Execute Addr:";e: LOAD *driv
e;n$CODE : RANDOMIZE USR e
1050 INPUT "Name:"; LINE n$: LOA
D *drive;CHR$ 0+n$
2000 GO TO 2000+option*10
2010 INPUT "Name:"; LINE n$: ERA
SE drive;n$: GO SUB 8000: GO TO
80
2020 INPUT "Name:"; LINE n$: ERA
SE drive;CHR$ 0+n$: GO SUB 8000:
 GO TO 80
3000 GO TO 3000+option*10
3010 LET drive=1: GO SUB 9200: G
O SUB 9990: GO SUB 8000: GO TO 8
0
3020 LET drive=2: GO SUB 9200: G
O SUB 9990: GO SUB 8000: GO TO 8
0
3030 LET drive=1: LET leng=1: LE
T factor=2: GO SUB 9200: GO SUB
9990: GO SUB 8000: LET leng=0: L
ET factor=1: GO TO 80
3040 LET drive=2: LET leng=1: LE
T factor=2: GO SUB 9200: GO SUB
9990: GO SUB 8000: LET leng=0: L
ET factor=1: GO TO 80
4000 GO TO 4000+option*10
4010 INPUT "Name of File:"; LINE
 n$: INPUT "New Name:"; LINE k$:
 MOVE 1;n$ TO 2;n$: GO SUB 8000:
 GO TO 80
4020 INPUT "Name of File:"; LINE
 n$: INPUT "New Name:"; LINE k$:
 MOVE 1;n$ TO 3;n$: GO SUB 8000:
 GO TO 80
4030 MOVE "d";1 TO "d";2: GO SUB
 8000: GO TO 80
4040 MOVE "d";1 TO "d";3: GO SUB
 8000: GO TO 80
5000 GO TO 5000+option*10
5010 RANDOMIZE USR 0
5020 RANDOMIZE USR 14070: GO SUB
 8000: GO TO 80
5030 RANDOMIZE USR 4007: GO SUB
8000: GO TO 80
5040 LOAD *1;b$
5050 INPUT "Drive Number:";drive
: GO SUB 8000: GO TO 80
8000 LET en=1: LET upl=VAL x$(wn
)+2: LET lwl=(VAL x$(wn)+VAL d$(
wn))-2: LET x=upl: LET y=VAL y$(
wn)
8001 PRINT AT x,y; PAPER 0; INK
VAL p$(wn); BRIGHT 1;" ";c$(wn,e
n,1 TO VAL w$(wn))
8002 IF INKEY$="" THEN GO TO 800
2
8003 GO SUB 8009: IF INKEY$=CHR$
 11 AND x>upl OR INKEY$="6" AND
x>upl THEN LET x=x-1: LET en=en-
1
8004 IF INKEY$="q" THEN FOR f=1
TO 21: PRINT AT f,0;"
                           ": NEXT f:
GO TO 50
8005 IF INKEY$=CHR$ 10 AND x<lwl
 OR INKEY$="7" AND x<lwl THEN LE
T x=x+1: LET en=en+1
8006 IF INKEY$="0" OR INKEY$=CHR
$ 13 OR INKEY$=CHR$ 12 THEN GO T
O 8100
8007 IF INKEY$<>"" THEN GO TO 80
07
8008 GO TO 8001
8009 PRINT AT x,y; PAPER VAL p$(
wn); INK VAL i$(wn);" ";c$(wn,en
,1 TO VAL w$(wn)): RETURN
8100 LET option=en: RETURN
9200 LET x=1
9201 FOR f=1 TO 20: PRINT AT f,0
;q$(1,1 TO ): NEXT f
9220 LET col=0: LET l=2
9230 CLOSE #5: OPEN #5;" CAT ";d
riveRND16,110
9240 FOR f=1 TO 110
9250 LET a=CODE INKEY$#5
9260 LET b=CODE INKEY$#5
9270 LET len=a+256*b
9280 LET a=CODE INKEY$#5
9290 LET b=CODE INKEY$#5
9300 LET strt=a+256*b
9310 LET a=CODE INKEY$#5
9320 LET b=CODE INKEY$#5
9330 LET end=a+256*b
9340 LET a$="": FOR g=1 TO 10: L
ET a=CODE INKEY$#5: LET a$=a$+CH
R$ a: NEXT g
9350 IF l>20 THEN LET l=2: LET c
ol=col+11: IF col>23 THEN PRINT
#0;"More?": GO SUB 9390
9360 IF end=65535 THEN GO TO 50
9370 PRINT AT l,col;a$: IF leng=
1 THEN PRINT AT l+1,col; BRIGHT
1;":";(256*(end-strt))+(len-1)-5
9380 LET l=l+factor: NEXT f
9390 IF INKEY$="n" THEN FOR f=1
TO 21: PRINT AT f,0;q$(1,1 TO ):
 NEXT f: GO SUB 9990: GO TO 80
9391 IF INKEY$="y" THEN FOR q=1
TO 21: PRINT AT q,0;q$(1,1 TO ):
 NEXT q: LET l=2: LET col=0: NEX
T f
9400 GO TO 9390
9987 STOP
9988 LET f=wn: LET w=VAL w$(f):
LET d=VAL d$(f)-1: LET x=VAL x$(
f): LET y=VAL y$(f)
9989 RETURN
9990 GO SUB 9988: LET f=wn: LET
p=VAL p$(f): PRINT AT x,y; PAPER
 p; INK VAL i$(f); INVERSE 1;q$(
1,1 TO w+1): FOR z=x+1 TO x+d: P
RINT AT z,y; PAPER p;q$(1,1 TO w
+1): NEXT z
9992 OVER 0: INK VAL i$(f): PAPE
R 7: PLOT (y*8)-1,175-(x*8)+1: D
RAW (w+1)*8+1,0: DRAW 0,-((d+1)*
8)-1: DRAW -((w+1)*8+1),0: DRAW
0,((d+1)*8)+1
9993 PRINT PAPER 7; INK VAL i$(f
); INVERSE 1;AT x,y+1;l$(f,1 TO
w-2);AT x,(y+w-4); INVERSE 0; IN
K 0; PAPER 2;"?"; INK 2; PAPER 6
;"?"; INK 6; PAPER 4;"?"; INK 4;
 PAPER 5;"?"; INK 5; PAPER 0;"?"
9994 FOR z=1 TO d-2: PRINT PAPER
 VAL p$(f);AT x+z+1,y+1;c$(f,z,1
 TO w-2): NEXT z
9995 RETURN
```

Figure 6

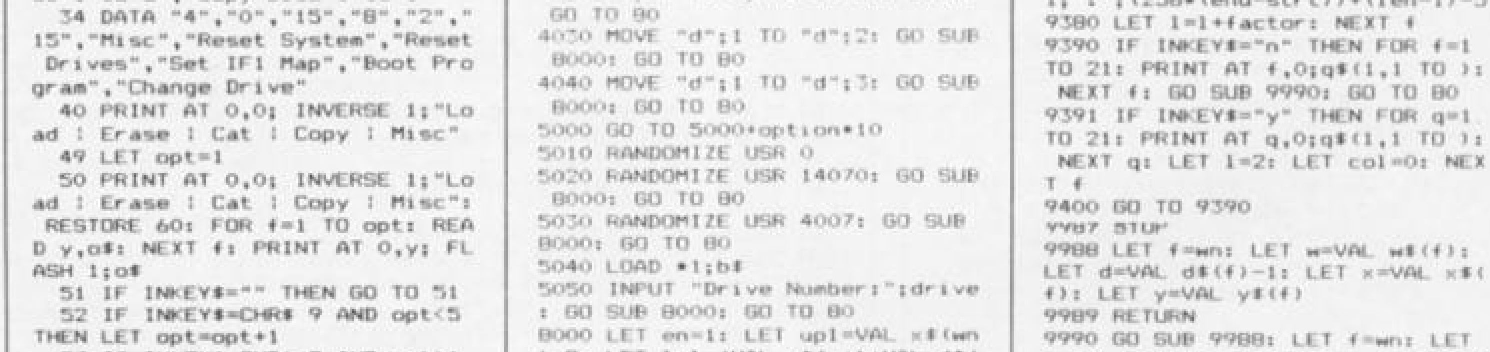


Figure 7

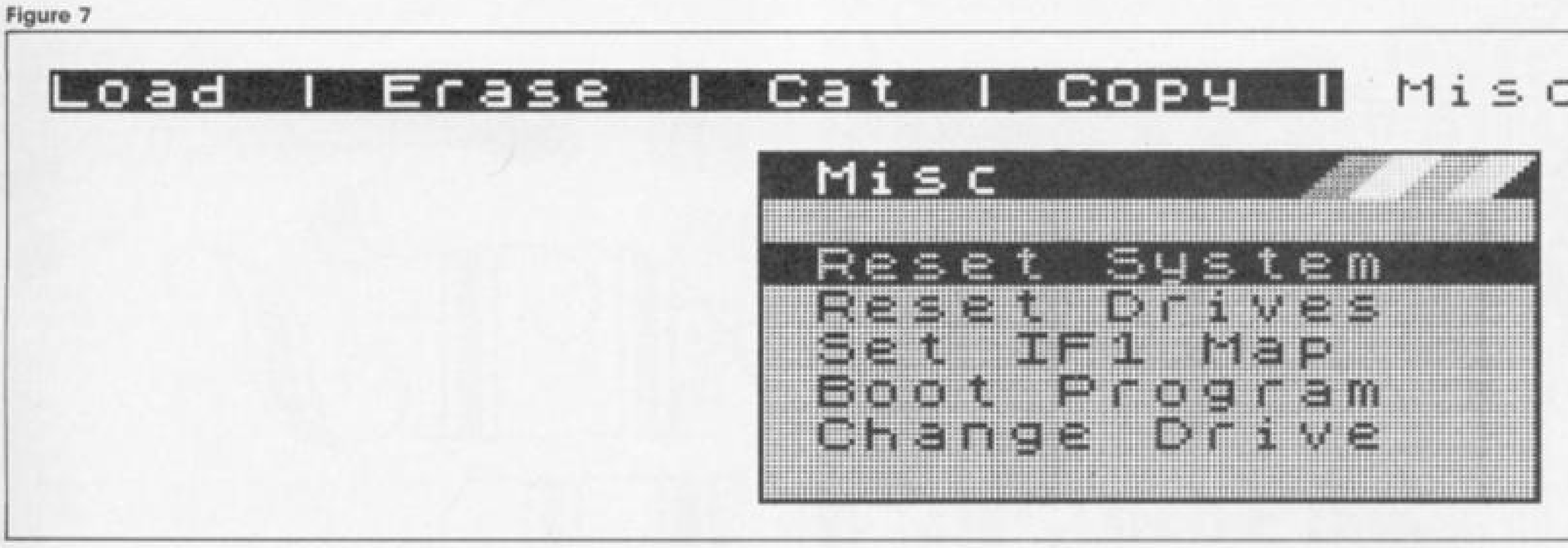


Figure 8

```
    5 REM
       ***RENAME PROGRAM***
       ***TOM  NICHOLSON***
   10 BORDER 1: PAPER 5: BRIGHT 1
 : INK 0: CLS
   20 LET A=1: LET B=15: LET K=2
   30 CLOSE #4: OPEN #4;" CAT ";1
 RND 16
   40 FOR n=a TO b
   50 POINT #4;n
   60 DIM a$(16)
   70 FOR f=1 TO 16
   80 LET a$(f)=INKEY$#4
   90 NEXT f
  100 IF CODE a$(k)=255 OR CODE a
 $(k)=229 THEN PRINT '" "; INVERS
 E 1;n-1; INVERSE 0;" IS THE LAST
  FILE ON DISC.": GO TO 130
  110 IF n<=9 THEN PRINT "  ";N;"
 ";: PRINT a$(7 TO 16): LET k=1:
 NEXT n
  120 PRINT " ";n;" ";: PRINT a$(
 7 TO 16): NEXT n
  130 INPUT ("    ReName=(1,2, et
 c.)"'"N=Next page S=Stop R=Re-St
 art "); LINE y$
  140 IF CODE y$=78 OR CODE y$=11
 0 THEN GO TO 320
  150 IF CODE y$=83 OR CODE y$=11
 5 THEN STOP
  160 IF CODE y$=82 OR CODE y$=11
 4 THEN GO TO 10
  170 IF y$="" THEN GO TO 130
  180 IF CODE y$(1)<48 OR CODE y$
 (1)>57 THEN GO TO 130
  190 LET y=VAL y$
  200 INPUT "New Name : ";n$
  210 POINT #4;y
  220 FOR f=1 TO 6
  230 LET a$(f)=INKEY$#4
  240 NEXT f
  250 FOR f=1 TO 10
  260 IF f<=LEN n$ THEN PRINT #4;
 n$(f);
  270 IF f>LEN n$ THEN PRINT #4;"
 ";
  280 NEXT f
  290 IF y>15 THEN CLS : GO TO 30
  300 CLS : IF k=1 AND a<16 THEN
 LET k=2: GO TO 30
  310 STOP
  320 CLS : LET a=a+15: LET b=b+1
 5: GO TO 30
```

THAT'S THE TICKET

Carol Brooksbank with advice on using screen dumps in a more versatile way

Every club should have among its members a Spectrum owner with a printer capable of screen dumps. Why? Because the Spectrum can take care of all the printing of tickets, posters, notepaper headed with the club logo — in fact, virtually all of the printing requirements can be handled cheaply and quickly by someone with a bit of imagination and a screen dump or two.

There is such a choice of printer/interface combinations these days that it is impossible to say that any one is better than all the others. I use the Wafadrive centronics interface with the Rotronics screen dump program Draw DX-85, and an Epson RX80F/T printer. The Wafadrive Draw program has the great advantage of offering three sizes of screen dump, and it also allows you, by poking values into the program, to take advantage of the Epson's variable bit-image graphics modes. This makes it possible to change the proportions of the screen dump in printing, subtly altering the width of the image on the paper. Obviously the more facilites of this sort you have, the more versatile you can be, but the most important thing is to be really familiar with your interface/printer, to know just what it can and cannot do.

Figure 1



A commercial graphics program can be a great help. I use Softechnics' The Artist, but, again, any similar package will do, or you can use a light pen or even BASIC or Spectrum block graphics.

Unless you are lucky (or rich) enough to own a colour-jet printer, your printouts are going to be in black and white, so

always work in black and white on the screen, and keep it simple. Fussy graphics may gain gasps of admiration on the screen. They usually look a jumbled mess on paper.

You need to know whether your screen dump gives a true or distorted reproduction, and the easiest way to find out is to draw a circle on screen and make a screen dump. Is it still a circle? If not, you will have to allow for the distortion in your designs. The choristers in the Choir Supper poster have round heads on screen, but the distortion in the screen dump gives a more pleasing proportion on paper (**Figure 1**). In the Laughter and Tears poster, the tape reels had to be round, so they are egg-shaped on screen (**Figure 2**) You must either take advantage of the distortion or compensate for it. Remember that it is the effect on paper that matters, and make plenty of test dumps as you go along.

Multiple dumps

There is no need to limit yourself to one screen dump per poster. Figures 1 and 2 each consist of three. In figure 1, the lettering is one dump in size 3, and the cartoons of choristers singing and eating are each a separate dump in size 1. In figure 2, two size 3 dumps one above the other make an A4 poster, with the gramophone, size 1, superimposed. If you have the facility for different sizes, small detailed sections are often more successful if they are drawn full size on the screen and reduced in the printing. The Laughter and Tears tickets were produced by printing the poster screens side by side, in size 1, on thin card, (**Figure 3**). The main motif is used again, size 1, as a logo on the programmes. (**Figure 4**). Juggling with the sizes and positions in this way lets you produce a number of related printed items, using only one or two screen designs.

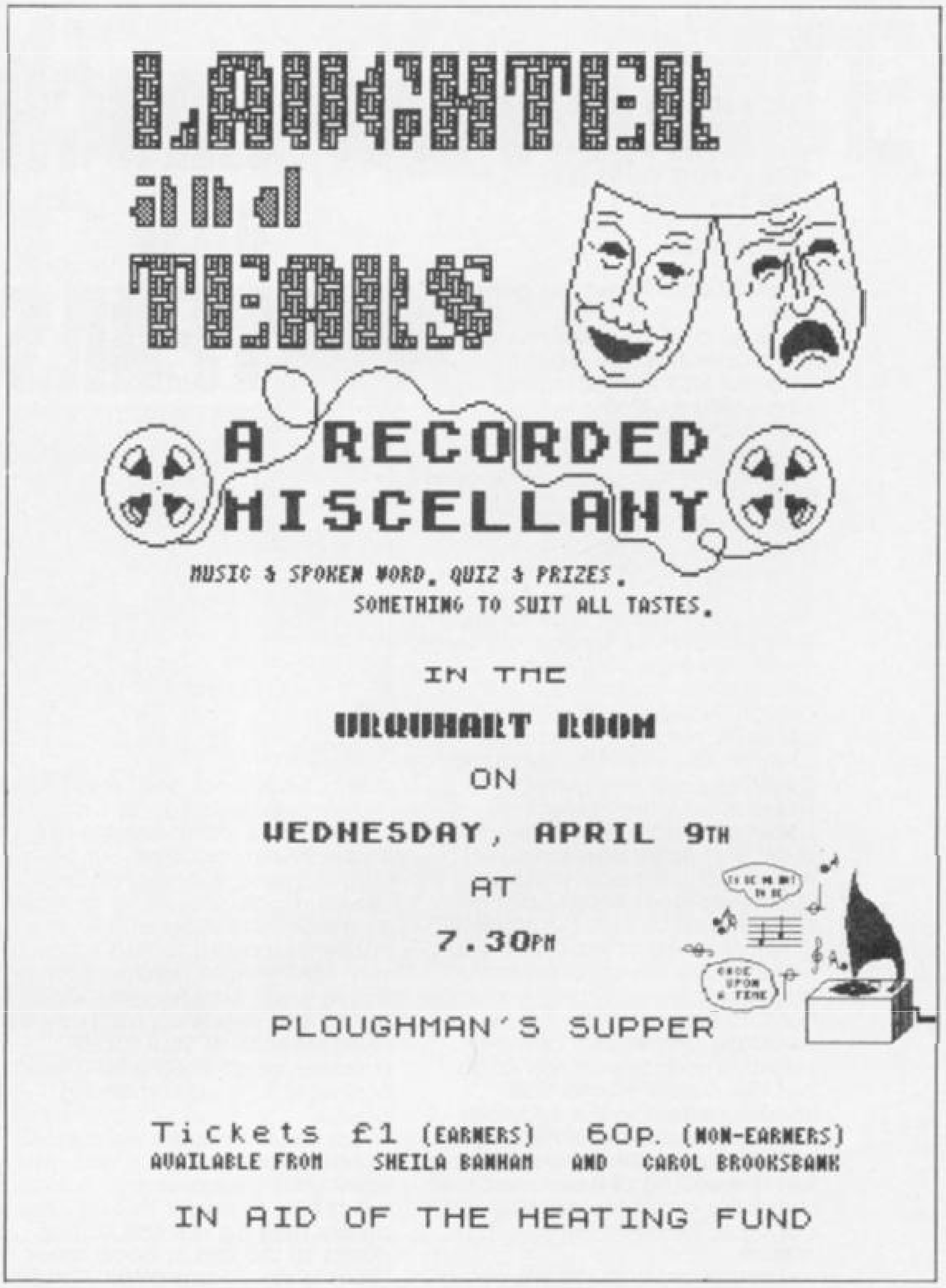
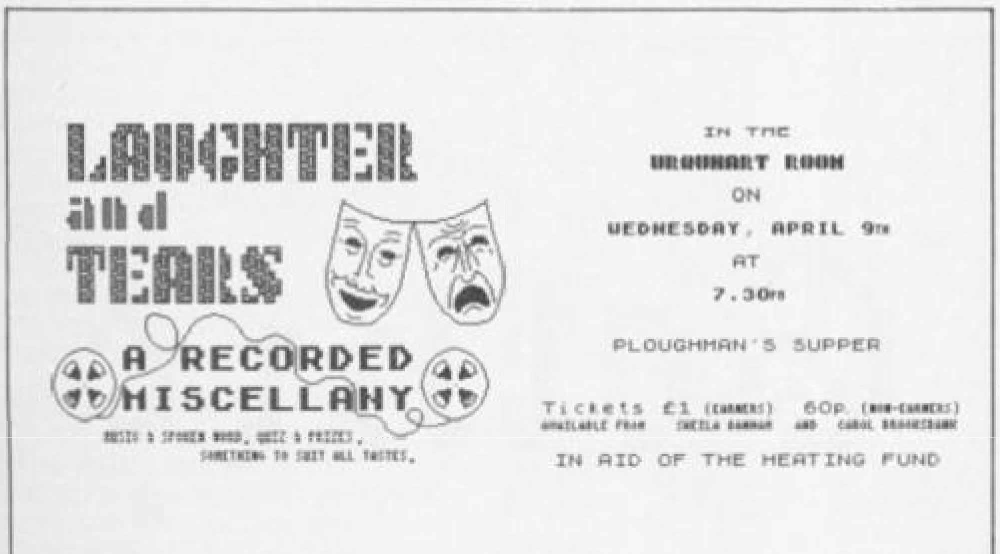


Figure 2

Figure 3



If you are using more than one dump it is vital to align the paper properly for each one. Find something on the printer which you can use as a reference point — I use the numbered bar which presses against the paper — and find out where this falls on the first dump when the second one is in the correct place on the paper. You will have to experiment until you get the effect you want, but once you have it, write the position of your marker down. Then, when you do your print run, you will know how to line the paper up.

My program for printing a poster run looks something like this:

```
10 LOAD "post1" SCREEN$
20 POKE 23296,3
30 GOSUB 1000
40 PAUSE 0
50 LOAD "post 2" SCREEN$
60 GOSUB 1000
70 PAUSE 0
```

```
80   LOAD "post 3" SCREEN$
90   POKE 23296,1
100  GOSUB 1000
110  STOP
1000 FOR x=1 TO 10
1010 DRAW *
1020 OPEN# * 3,"C"
1030 LPRINT CHR$ (12);
1040 CLOSE# * 3
1050 NEXT x
1060 RETURN
```

Lines 20 and 90 set the dump size.
Lines 40 and 70 allow me to put the paper through again, aligned for the next dump.
Lines 1000 to 1060 are the subroutine which prints the required number of copies.
Line 1010 is the Watadrive screen dump command.
Lines 1020 — 1040 send a form feed to the printer.

Lettering is very important in most printing work. The Artist offers a number of type faces, and the 'overlay' option allows you to enlarge, reduce or change the proportions of the design. However, I find that this option is not really accurate enough for lettering, because distortions are sometimes introduced which may vary between instances of the same letter, so I have developed my own machine code program which replaces each pixel in the original letter by a block of pixels in a specified size. If you have to use the enlarge option in a graphics package, you may have to tidy the lettering up, because differences between letters in your poster will stand out like a sore thumb. The shading effect in the Laughter and Tears poster is obtained by hollowing out the letters, leaving just the outline of each one, then using the graphics package 'fill' option to fill each letter with a texture.

Experiment with type faces to make sure they still 'work' in the size you plan. The Artist gothic font looks fine on screen or in size 1, but enlarged too much it is terrible. (**Figure 5**).

Colour

Even with a monochrome printer, there are tricks you can use to make your posters more eye-catching. Printing on coloured paper can be effective, but your printer will need a cut paper feeder, because I have never come across tractor feed paper in colours. Ribbons are available in more than one colour for some printers. For the Epson, my local dealer stocks red and blue as well as black, and the cartridges are easily changed, so the dumps can be in different colours. I was interested to see a review in the November ZXC of a new program — Poster Machine — which lets you produce large posters in sections from a screen dump. I have not tried it yet, but it sounds promising.



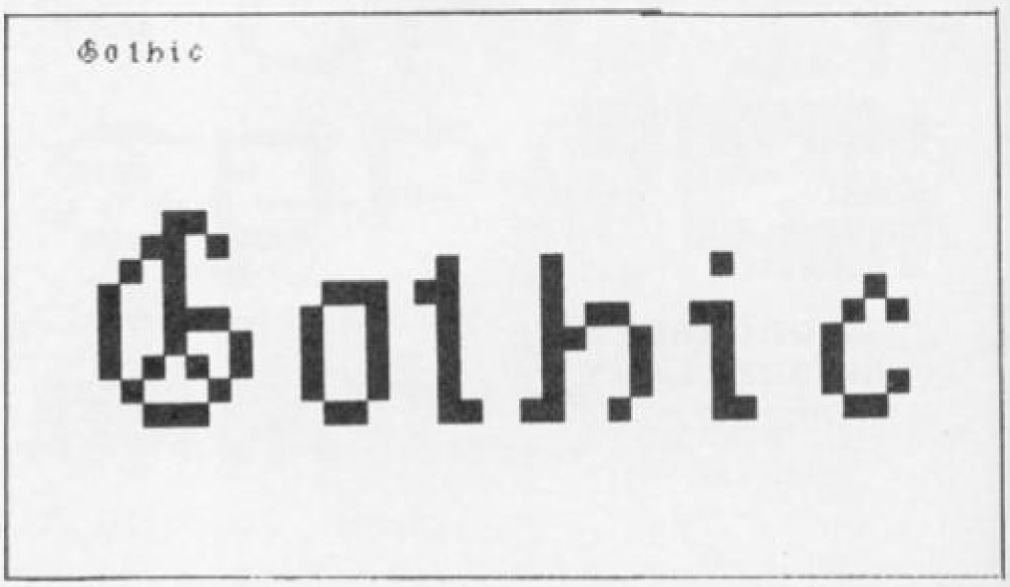
Figure 4

For tickets, card can be used if your printer accepts cut paper, but my Epson will only accept very thin card. Anything as thick as a postcard sticks in the roller. I have managed to find some very light weight card in A4 size which it will handle, and I can print three tickets on each sheet. Investigations at your local stationer or art shop will probably turn up something similar.

Logos for headed notepaper are the easiest of all. Once you have your design screen, simply run a box of paper through the printer making a small screen dump at the top of each sheet. There is very good quality tractor feed paper around — my favourite is Blue Chip — so you can write a FOR-NEXT loop with a screen dump and a form-feed, set the thing going and leave it to print the whole box while you walk the dog, go to the pub or do the Times crossword.

Any club, church or group which needs printing, especially for fund raising, will find your services invaluable. If you have a small business, printing your own stationery with your logo and heading will not only save money, but also avoid disaster when you find you have almost run out of some vital form, and have not ordered a fresh supply. Once you have the design screen, the Spectrum can run the new ones off in no time. You do not need to be an artistic genius. Simple designs are often more effective than elaborate ones. Next time your club is having a function, try designing a few tickets or posters and showing them round. You will find it is great fun to do them anyway, but don't blame me if you never have any spare time afterwards, because you are always in demand as a printer.

Figure 5



BECOME A FORTUNE HUNTER!

Have you got what it takes to suffer the slings and arrows of outrageous football fortunes? Prove your footballing knowledge in our quiz and you could carry off CDS's new football game.



Brian Clough's Football Fortunes puts you in the managerial hotseat and confronts you with the problems that face real life club supremos, such as injuries to star players, cash flow crises and team selection.

Football Fortunes is an absorbing blend of computer and board game and your success depends on building up a strong team. Your players are represented by cards bearing the names of well known footballers. Each footballer has a "star rating". The bigger your total team star rating the better your form.

You can dabble in the transfer market and try to pick up highly rated players by bargaining with your opponents. However even if you assemble a brilliant team there are enough pitfalls built into the game to guarantee that you can't be sure of victory until the proverbial final whistle.

Above all Football Fortunes is designed to be a sociable game (for 2 to 5 players) and it's up to you whether the tactics used to win league and cup honours will be "hard but fair" or studded with "professional" fouls.

The game will have an instant appeal for the soccer fraternity but you don't have to be a football fanatic to enjoy it.

How to Enter

Write your answers on the coupon provided and send your entries to Football Fortunes Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.

Soccer posers

There are 15 copies of Brian Clough's Football Fortunes to be won and all you have to do is answer three simple footballing questions.

1) Who are the current holders of the World Cup?
2) Which was the last team to win the League and FA Cup double?
3) Who is the manager of Barcelona?

The competition is open to all readers of ZX except employees of Argus Specialist Publications, Chase Web and CDS.

The editor's decision is final and no correspondence can be entered into. Please remember to write your answers on the back of your entry envelope. The closing date is March 6th.

Football Fortunes Competition

The answers are,

1) ....

2) ....

3) ....

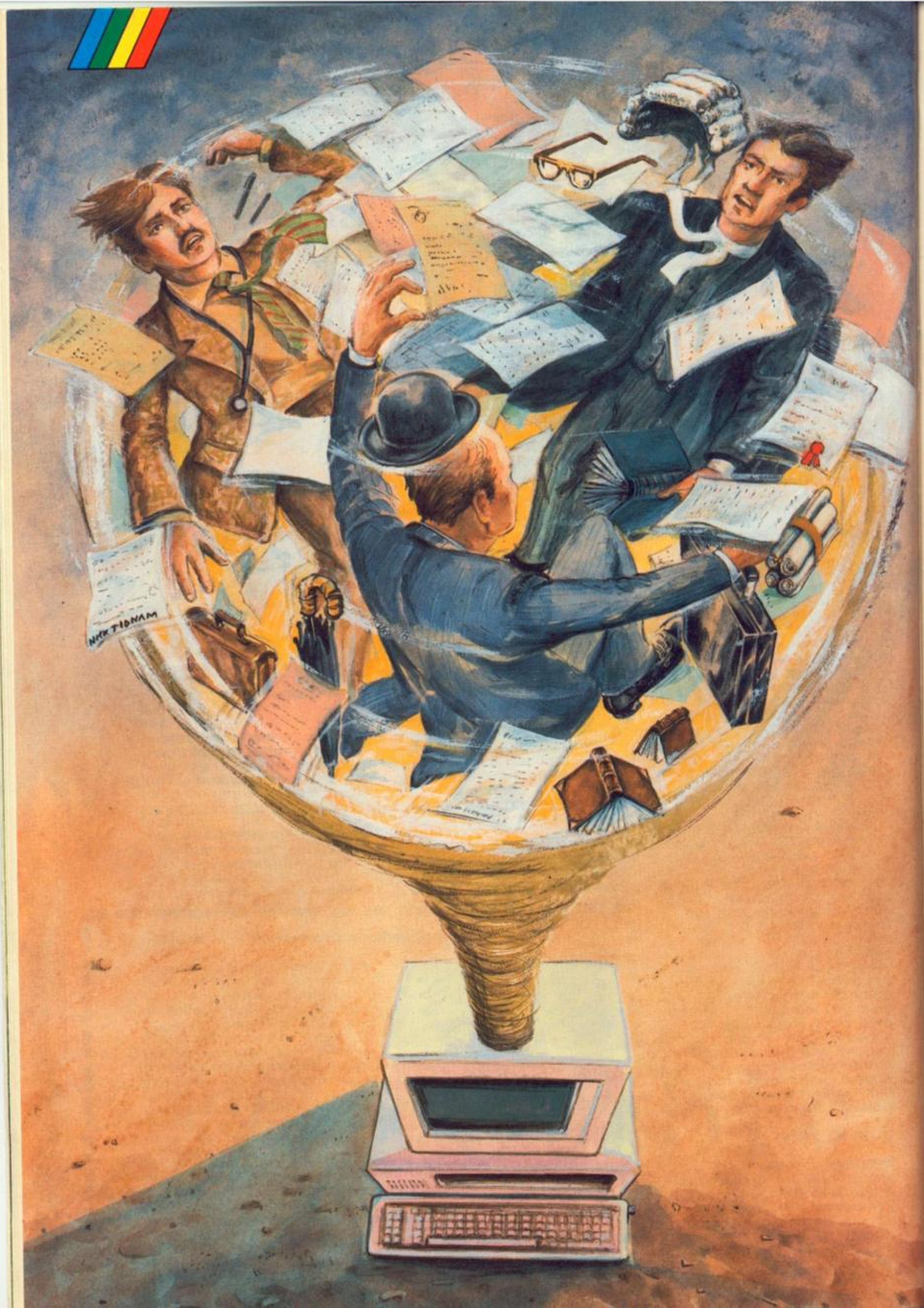
Name: ....

Address: ....

....

....

Send your entry to Football Fortunes Competition, ZX Computing, No 1 Golden Square, London W1R 3AB.


NICK TIDNAM

EXPERT SYSTEMS

Would you accept advice from your computer? David Nowotnik introduces a new series that will show how the principles of expert systems can be applied to the Spectrum and QL.

A few years ago, a popular series of TV advertisements for one of the major banks had the theme of a 'personal' bank manager living in a customer's wardrobe. Do you remember it? These advertisements tried to put across the idea that bank managers (of that particular chain) were more approachable for financial advice that commonly envisaged. Despite that advertisement, bank managers continue to seem as distant as before. But before very much longer, many homes could really find a personal financial adviser in the wardrobe, or somewhere more convenient in the home. That advisor could be their own personal computer; a computer with human-like expertise.

For years, one principle direction of computer science has been the development of programs which mimic in some way human behaviour. The field generally is called artificial intelligence; AI for short. Popular science fiction has predicted what AI might be achieving for us in the future. Computers with intelligence equalling or surpassing that of humans has been the subject of several popular movies and TV series; remember HAL in '2001' — R2D2 and C3PO in 'Star Wars' — ORAC of 'Blake's 7 — KITT in 'Knight Rider'? All these are computer systems which perform tasks such as give advice, assist, inform, plan, forecast, and diagnose.

Naturally, computers have not yet achieved the level of sophistication represented in futurist fiction, but computer programs have been developed, and will continue to be developed, which begin to approach this level of AI. The ability to place human expertise into a computer program is just one branch of AI programs which achieve this are called 'Expert Systems', and, believe it or not, expert systems are already in use in science and industry. It's only a matter of time before they'll be entering our homes. They will be our personal 'bank manager'. They will also give us advice on a variety of subjects; health, education and law, to name a few.

The Turing Test

The concept of AI and expert systems has been with us for some time. Even before electronic computers were on the scene, concepts of AI were being considered. Alan Turing, whose tragic story is now the subject of a West End play, gave his name in 1936 to the Turing Test. The ultimate test of AI would be for a human to sit at a terminal and not know whether he was communicating with another human or a computer. Later, in the 1960s, scientists worked on the idea of the 'General Problem Solver', the ultimate computer system which had the intellectual capability to solve all problems. While this wonderful idea failed, this work gave rise to some of the principles used in the 1970s and '80s to generate programs which were able to capture and utilise human knowledge and experience.

Development is still going on; the Japanese have made great strides forward in AI with their work in the development of the fifth generation computer. One of the aims was to develop systems which have that most human of qualities — common sense. The sheer complexity of that problem has been relegated to the next generation of computer systems!

To understand how an expert system works one has to analyse how experts work. Human experts have a wide knowledge of their subject. But knowledge is just one part of being an expert. For example, there must be a good reason for a company to pay a senior accountant more than a recently qualified graduate accountant. The graduate should have a broader and fuller knowledge of accountancy than his senior colleague. The senior man certainly will not have the same knowledge of up-to-date methods and techniques as the graduate. But what the senior man has over and above the graduate is experience. Experience is that ability to judge what knowledge and information is necessary to make a decision, and what importance to attach to various pieces of information.

Advice

We don't become experts simply by reading books on a subject. All we have is knowledge; we need to develop experience to use that knowledge to make decisions. We may read a text book on, say, stocks and shares, but we still need to go to an advisor to find the best time to sell our British Telecom, TSB, or British Gas shares. But buy and sell shares only a few times you soon start to gain the experience necessary to make your own decisions on transactions.

So, an exert system has to be more than just a database; more than a 'book' of knowledge. Many readers will be using databases on their Sinclair micros. In using your database, you are providing yourself with information. You make the decisions on which information to retrieve, and you make judgements and decisions on the data presented, on the basis that you know and understand the importance of the information, and its relevance in making a decision. In the expert system, it is the computer program which selects the information and makes judgements according to rules presented to it. These rules are the same as a human expert would apply, sometimes sub-consciously. To produce an expert system a specialist called a knowledge engineer will interview an expert to extract those rules from him, so that they can be built into a computer program alongside the database.

It is perhaps a little misleading to liken the knowledge built into an expert system to a database. It is more usual to link that knowledge in some way to the rules. There are many different ways of building an expert system, but one of the most common is the rule based system (RBS). Here rules, very much like IF . . . THEN statements of BASIC form an integral part of the knowledge base.

But to determine which rules should apply to a particular problem, another section of program has to be added; this is called the inference engine.

EXPERT SYSTEMS

Think of it as an index of rules; a database in which the rules are stored, as well as features of the rules. As in a database, the fields in which these features are stored can be searched, and particular records (rules) selected and applied to any problem in question.

The final part of the expert system has to be the human interface. Just like dealing with a human expert, the expert system must determine what it is expected to achieve (its goal), then ask the right questions to provide itself with the information to match against its knowlege base. It has to be able to explain its conclusion in a form which any non-expert user can understand, and it must be able to explain why it reached a particular conclusion.

Diagnosis

All that is a lot to ask of a 'dumb' computer, and it is little wonder that expert systems can take thousands of man hours to develop. Most of the early systems were custom built. The earliest system of any significance was called MYCIN. Developed in 1976, it contained over 400 rules and provided advice to doctors on the diagnosis and appropriate antibiotic treatment of blood bacterial diseases. While this program was a milestone in expert system, it also provided important lessons in user friendliness and in the treatment of uncertainty.

Normally, an IF . . . THEN rule will provide a certain result if a condition is met. In BASIC, a line

IF x=1 THEN LET b=2

will always cause b to become 2 whenever x=1. That is fine in mathematics, but what of real life. Say you had an expert system in which you wanted to determine the species of a farm animal. Then:

IF it produces milk THEN it is a cow

might be one rule. In most cases this would be correct, but goats could also be milk producers. So there is a chance that 'if it produces milk it is not a cow'. There is some uncertainty in the rule. Most human experts in making decisions have to deal with uncertainty. It is rare that all the information necessary to make a certain prediction of an outcome. Human experts will deal with such uncertainty in an empirical way. They will, unconsciously, be applying probabilities to rules (gained from experience or knowledge), and combining these probabilities in some logical fashion to provide degrees of uncertainty on various outcomes.

For example, a stockbroker will make a judgement based on knowledge and experience on whether a share price will rise or fall. There is uncertainty in the judgement, but an experienced man should be able to pick most likely stocks to rise or fall.

In the milk producer example, if, say, 98% of milk produced comes from cows, then there will be a 98% certainty on the rule 'if it produces milk then it is a cow' being correct. In expert systems, statistics can be used to combine the probabilities from various rules more precisely than the human expert in predicting an outcome. In building the expert system, the expert, in many cases, must provide a certainty factor with each rule. The computer will then calculate very precisely the overall probability in coming to a decision. Of course, what it still lacks is common sense!

As I said at the beginning, it won't be too long before expert systems will become available on home computers. Arguably, they have arrived already. Digital Precision have recently announced the 'Better Basic Expert System' for the QL. At the time of writing, no review of this product has appeared, so there is no confirmation of the supplier's claims. But, it appears that this program will automatically scan a SuperBASIC program and improve its style and correct mistakes. The human equivalent would be the expert programmer who corrects the efforts of a novice. Because of the complexity of the problem, Digital Precision's product deserves to call itself an expert system — if it works as promised.

Shells

Writing a custom-built expert system from scratch does require an enormous amount of effort. To simplify matters, there are programs recently available for business microcomputers which form the basis of an expert system. These programs are called 'shells'. In most shells, you, the 'expert' provide the rules and, perhaps, the certainty factors. In others, the program learns by example. You provide the outcome and the factors which influenced the outcome, and the system builds rules based upon these observations. This is very much like the way a human expert will have built his expertise. The ability for an expert system to be taught new rules is important if the field of expertise can undergo change — if the rules which govern decisions can change.

For business, expert systems have a logical place. For instance, staff losses, particularly at senior levels, can result in major problems. Place that expertise on a computer, and the replacement to the departed member of staff will have a good basis for continuing from where the previous incumbent left off. Certain company experts, for example accountants, are frequently called on to give advice. They may not always be readily accessible to those wanting advice. Place their expertise into an expert system. That program can be copied many times over, and distributed to the people that need it. This, hopefully, doesn't mean that human experts can be replaced once they have committed all to an expert system, just that they will be called on when necessary. After all, common sense is still necessary in many cases.

Experts at home

But what about home users? Probably the ordinary man at home is more in need of expert systems that the man in a business environment. Frequently, he doesn't know who to approach to get advice. The recent surge of interest in share ownership with the British Telecom, TSB and British Gas share issues has increased dramatically the number of people in this country owning shares. Filling in the forms to acquire shares was easy. But how do you dispose of them, and when, and what of other shares? And where do you go for advice on such matters? I wouldn't be surprised if an expert system for this application appears soon — for house purchase and the diagnosis of simple ailments can't be long away either. There's a market there for any budding entrepreneur!

Hopefully, you have reached this point with something of an understanding of the nature of expert systems, and the basics of how they work. If you want to take that knowledge a bit further, then over the next few months I'll be explaining the principles in a little more detail with simple examples for you to type in and try on your Spectrums and QLs. LISP and PROLOG are the more popular programming languages of AI, but it is surprising what can be achieved in BASIC. Well, hopefully you will be pleasantly surprised over the next few months!

QL news from Brian Beckett

THE LAST ZX microfair of the year was held on 13 December and you might be forgiven for thinking that it was the Spectrum that was terminally ill rather than the QL. For a pre-Xmas fair, hardly anybody was there (maybe no one's giving presents this year) and virtually all the Spectrum stuff had been seen before. A very notable exception is Prehistoric Adventure (£9.99) from Crusader Computing. It's an adventure game which takes the hero "Ohio" (remind you of anybody?) from Stonehenge into a prehistoric world on search of the elixir of life. He faces various pitfalls and dangers including loads of unfriendly dinosaurs. It's a very good and well designed game and you get a free dinosaur poster that's very nicely done and — if you find the going too tough — you can send off for a hint sheet.

QL COLUMN

Apart from this and a couple of other Spectrum games, the show pretty much belonged to the poor old QL. The big news for you quantum leapers out there is up-coming keyboards for those sick and tired of trying to do serious work on your original black box. The Schön QL keyboard due for release around Christmas will sell at £54.95 and just slots in to replace Sinclair's original. So it's compatible with all existing peripherals and the prototype that I played with had a good professional feel to it. Apart from the function keys (which are in red), the replacement board is in good-old-QL-matching basic black and the overall result is a QL that takes on an Amstrad-like appearance.

But for those of you who want to fool your friends into thinking you've tossed the old QL out and bought something in the IBM price range, those devilishly clever West Germans at ABC Elektronic are preparing to release a QL upgrade kit consisting of a fully separate IBM-format, XT-style, 83-key keyboard and a box for the main board. The kit will sell at £210 and will be available in the UK from Digital Precision. For your £210, you'll get the keyboard, the mainboard box, two 3.5 disc drives (or one 3.5 and one hard disc drive), a genuine on-off switch and two expansion slots. If you prefer, you will be able to buy the keyboard by itself for £99.95 which is fitted with a special interface slotting into the processor socket and fully user installable without dragging out the soldering iron. This just gives you another peripheral hanging out of the old black box but it's a lovely keyboard and well worth considering.

Buying the full upgrade kit, however, will turn your very own quantum leap into something that looks like an IBM and — with some added internal memory — it should even behave like one save for the fact that neither God nor ABC Elekronic can make it IBM compatible. Remember that diehard QL owners will soon be faced with hordes of boring people bragging about their new Amstrad PCs and from what I have seen so far this upgrade kit should give plenty of scope for putting them in their place. When the kit becomes available, I will report in greater detail.

Raider II

I would like to send my special thanks to the long suffering staff of Microdeal who were forced to endure my nine-year-old endlessly playing the firm's new QL game Stone Raider II at the company's Microfair stand. This is the game Microdeal promised us for Christmas and (as my son can testify) it's a good one with lots of things to block, kill or otherwise frustrate you in your collecting of hexogems. It sells for £19.95 and sadly it looks to be the last QL game from Microdeal unless somebody comes up with a real winner. It's also likely to be Microdeal's last ZX Microfair since, like everybody else's, their stand wasn't very busy this time around and the QL is not longer a profitable area of the games market.

Another good new QL game is Tank Busters (£14.95) from Stellasoft. The object is to recapture the island of Stanley from the invading forces of one General "Galtihairy" in a series of tank battles. As tank commander you track and destroy the enemy by means of a sophisticated radar and an intelligent gunsight. The graphics and weapons simulation look good. Omega (£14.95) is a 3D arcade adventure coming on two microdrives (your QL needs some extra memory to run the game) involving a seek-and-destroy missin against a well-protected underground computer complex. The game is very involved, entertaining and a real challenge. I suspect most QL owners have some extra memory by now (or are about to get some as there are more and more bargains about) and designing games demanding it enables the programmer to create truly involved and highly sophisticated packages.

If you don't like copy-protected microdrives that won't let you fully duplicate the master (which, with those ever temperamental microdrives, is a constant worry), Compware has released Copycat at £10.99. It will duplicate the master of many old QL favourites and is sold under the strict condition that it will not be used for any illegal purpose (so behave yourself).

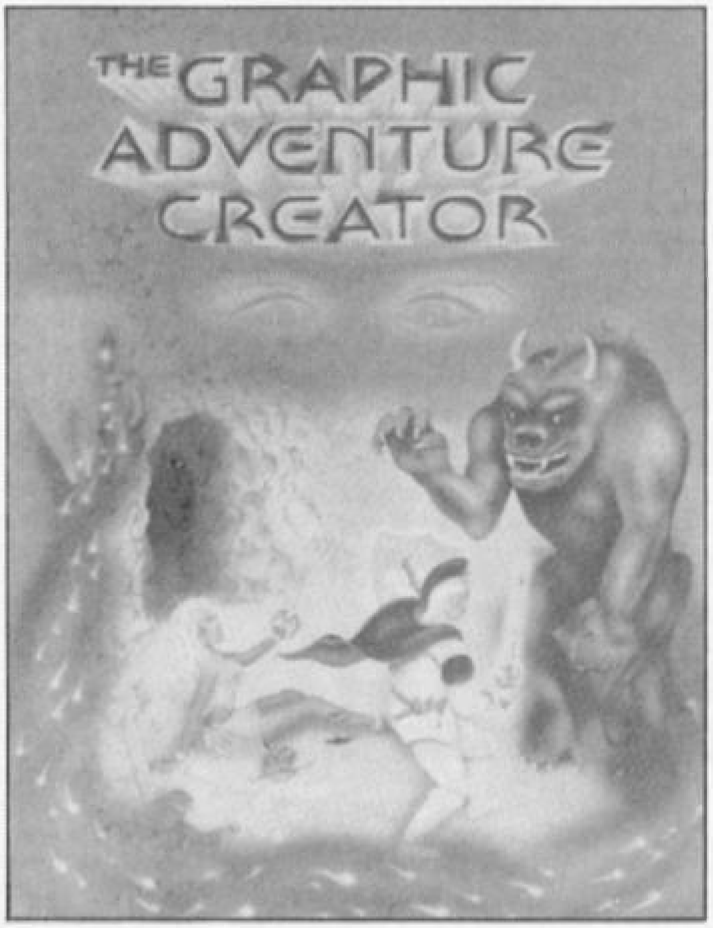
If your Ql has at least an extra 256K RAM (and you have a spare £25 left over from Christmas), get Taskmaster from Sector Software. It's a multitasking program which enables you (with sufficient RAM) to run all four Psion programs at the same time. The program contains a facility for backing up, copying and editing files and a calculator which enables the user to enter results directly into the particular program from which it was called. The idea is to set the QL separate tasks, eg. file sorting and printer driving, which the machine carries out at the same time. It is also possible to use Superbasic while multitasking with Taskmaster. It is a very sophisticated package and as long as you have enough memory it's possible to use it to multitask up to nine programs at one time. If you use your QL and Psion packages for complex and serious work, Taskmaster is an unquestionable bargain.

Liberator

I also picked up a Liberator (£60) Superbasic compiler from Liberation Software which is aiming to take on Supercharge in a big way. A factor of eight is described as "typical" for the speed-up in run-times for programs compiled with Liberator, and with large programs factors of up to 50 are claimed. It's a well-designed package with a good manual and the program has an assembler interface to support Superbasic extensions, procedures and functions written in assembly language. If you have an unextended QL, large programs may be compiled in stages. I don't have enough space to report Liberator's other features this month but I will report more next time.

A BACKWARD *Glance at Utilities*

Some utilities only show their true colours after a long period of use. Alan Davis reflects on the strengths and weaknesses of some established products.



The world of computing — as we all know — is a continually changing one. Hardly a month goes by without the appearance on the software scene of several new utility programs of one kind or another — and most of these are reveiwed pretty thoroughly in magazines like ZXC. In this way we're all kept up to date with new utilities, and get a good idea of how useful to us they're likely to be. Yet in some ways the reviewer's situation is artificial. Because of the pressure of copy deadlines, and the sometimes inevitable delays in the arrival of review samples, a reviewer may have only a few days to assess a program before writing his comments on it.

Fortunately ZXC is blessed with a group of excellent reviewers whose advice is indispensible — but the fact remains that in many cases some of the strengths (and weaknesses) of a utility program only emerge after extended use over a long period of time. And so I'd like to try an experiment in this article, of glancing back over several years of Spectrum programming to see just how some of the utilities I've personally used have performed in the long term (some have become nearly as valuable as my right arm, whereas others languish in a corner, unused and gathering dust).

Just how useful this exercise might be depends of course on the context. So let me say at the outset that virtually all my own use of the Spectrum falls into one of three categories:

(1) Adventure games
(2) Word-processing
(3) Scientific work, mainly involving statistical computations.

This does mean that if your interest lies mainly in programs like, say, the Argus "Arcade Creator" then I can't help. Also I tend to find my programming needs are often best served by writing specifically designed utilities myself for whatever job is in hand. That way you do get exactly what you want, rather than someone else's idea of what you MIGHT want. Even so, that still leaves me with quite a few commercial utilities to comment on. So let's start by looking at a couple of golden oldies. One has become indispensible, and the other proved a waste of time.

Compilers

One of the great temptations for the BASIC programmer is the idea that a BASIC Compiler could give him some of the advantages of machine code without the hassle of learning it. Like many others, I tried it — and it didn't come off. At the time when I was interested there were really, I think, only two such utilities under consideration — a "floating point" compiler by **SOFTEK**, and **MCODER 2,** by PSS. The Softek program was rather more versatile, but realy offered only a small improvement in speed compared with BASIC, and so I opted for MCODER, which only handled integers but offered a very considerable advance in speed of execution. Frankly, it proved more trouble than it was worth, and I never did write a complete program of any kind using it. The utility itself does its job excellently — but since the number of BASIC commands it can handle is limited, you really have to write your program specifically with the compiler's limitations constantly in mind. Generally, I found that this meant writing very inefficiently. Memory was rapidly gobbled up which — coupled with the memory occupied by the compiler itself — precluded the writing of a really large, full scale program and greatly limited the scope of what could be done. On the whole, I'd say that you'd be much better spending your money on a decent book on machine code.

Graphics

My other "golden oldie" is **Melbourne Draw,** and it's still the only commercial graphics utility that I use. Of course there are much more sophisticated screen designers around these days — the Rainbird "Art Studio" for example makes Melbourne Draw look very primitive indeed. But in fact, all I ever needed from this sort of utility (and my requirements haven't changed) was a sketchpad for trying out ideas on-screen, and for designing the occasional loading screen. For this, the program is excellent. It offers full control of the cursor in 8 directions, a fill routine, two

scales of screen magnification, and — importantly — a switchable character square grid which enables you to anticipate and solve attribute problems in your finished design.

Much of its appeal lies in the fact that it's so very simple to use, and so easy to customise if you want to extend its facilities for some special purpose. No gimmicks, no frills — just a good solid workhorse that I've grown to depend upon.

Assemblers

One utility which no programmer can do without is of course an assembler. At the time when I was looking for one, HISOFT'S **"DEVPAC"** was almost universally acknowledged as the best, and I've found it an immensely powerful tool which I use more than any other program. The package contains two microdrive compatible programs — an assembler and disassembler/monitor which can be loaded in separately. The latter contains a debugging facility which enables you to single-step through a machine code program, with the contents of registers displayed and continually updated on screen — incredibly useful when you're trying to hunt down that intractable bug which has eluded you. I've now used this package so much that handling it is almost second nature, and I'm therefore unlikely to change to another — but I do have certain reservations about its user-friendliness. The manual is very comprehensive, but can be somewhat confusing if you're just making a start on machine code (as I was when I bought it). Editing facilities are extensive, but rather cumbersome, and it does take some time to become used to the system. This complexity may well slow down the progress of a beginner, and if I were starting again from scratch, I think I'd be taking a close look at systems with more friendly editing facilities like Seven Stars' **"GENER 80/Moder 80"**.

Adventure Makers

I doubt if there's an adventure-writer in the land who hasn't had a dabble with either **"The Quill"**, or the more recent **'Graphics Adventure Creator"** — or both! A great deal of praise has been heaped upon the Quill over the years, even though some of the adventures written with it have been disappointing, and this is quite right. If you can't write a good traditional adventure with the Quill, then probably you can't write a good adventure, full stop! However, I must admit that it isn't a program I use these

This is an example of text printed with the QUALITAS "piazza" font. Four other fonts are available, as well as a font editor for designing particular characters or a whole new set.

This is an example of text printed with the QUALITAS "clarion" font. Note that this is a proportional font which gives a much more attractive appearance to your page.

Two examples of copy using Qualitas

days. I've written one full-length pure text adventure with it, which was great fun to do; but since my adventuring interests are inextricably concerned with developing interactive characters, the Quill doesn't really give me enough scope for what I want to do. Please note that I'm not knocking it; faced with superb Quilled programs like the St. Bride's "Snow Queen" (which gets my vote for the most delightful adventure of 1986) I'd be foolish to do so. It's just a case of horses for courses.

Incentive's Graphic Adventure Creator is perhaps a classic example of why an article like this may be of value — and I'll be very glad to have some feedback from readers on this. When I first saw this program on release, I was overwhelmed by the sheer power of it — and I still am. But there's a snag; a snag, moreover, that only becomes apparent after an extended period of trying to write a full length adventure with it. The problem is memory, or rather , lack of it.

After some considerable time and thought, I developed an adventure plot of some complexity which seemed admirably suited to the GAC, and set to work. A fortnight later I abandoned the attempt in despair. I had a few not-particularly-complicated graphics, about 15 locations, a fully operational beginning to the adventure . . . and only 5K left to finish it! Obviously this was a hopeless situation. It could be solved of course by writing the game in several separate modules, but I seemed to be able to get so little into a single module that the whole thing was likely to become impossibly cumbersome. This must, I think, put a very big question mark against the suitability of the GAC for anyone who wants to publish adventures as opposed to writing them merely for fun, and I'll be very interested to hear how others have fared with the program.

Word Processors

So much, then for adventure utilities. What about word-processing? Well, even when my only printer was a Seikosha GP 50 (which is a sort of dot matrix equivalent of the old ZX printer), I found **Tasword 2** extremely useful, even though the final copy was suitable only for your own use. Now that I have a full size printer, however, Tasword has joined the elite group of indispensibles! I've taken a look recently at both **"The Writer"** and **"The Last Word"** (two of its rivals) but I see no reason to change. Although these two offer more extensive facilities, I don't personally need the extras ("mailmerge", for example) — and I found both of them, particularly "The Writer", rather cumbersome to use. Perhaps I've just used Tasword for too long.

Qualitas

But this brings me to my most recently acquired utility — which alone gives me an excellent reason for sticking with Tasword. In October's ZX Carol Brooksbank reviewed **"QUALITAS"**, a program from Seven Stars Publishing which is designed for use with Tasword to produce a variety of near-letter-quality fonts on a dot matrix printer. Now I was already quite content with the NLQ of my Amstrad DMP 2000, but a facility to switch fonts, and even design them, sounded pretty useful. So I parted with my 7-95, installed QUALITAS (which is very simple) and proceeded to boggle at the results. Frankly, if someone had told me that this kind of quality was possible on a cheap dot matrix printer, I'd have laughed at them. Gone for ever is that rather mechanical, "computerised" appearance which my friends complain of in my letters. The price you pay for the beauty of the result is a rather slower printing speed — but who cares? If you have a dot matrix printer and Tasword, then rest assured that this is one utility whose purchase you certainly will not regret! (But do check with Seven Stars on printer compatibility first, at 34 Squirrel Rise, Marlow, Bucks, SL7 3PN).

Well, there you are. A motley bunch of utilities, some of which have made a pretty major impact on my programming life, one way or another. If you've had lengthy experience of others, or if you disagree with my assessments of any of these, then why not drop Bryan, the editor, a line and let us all know? There must be a great deal of this kind of experience among the readers of ZXC, and here's an excellent way of pooling it together so that we can all benefit. What do you think?

GAMES

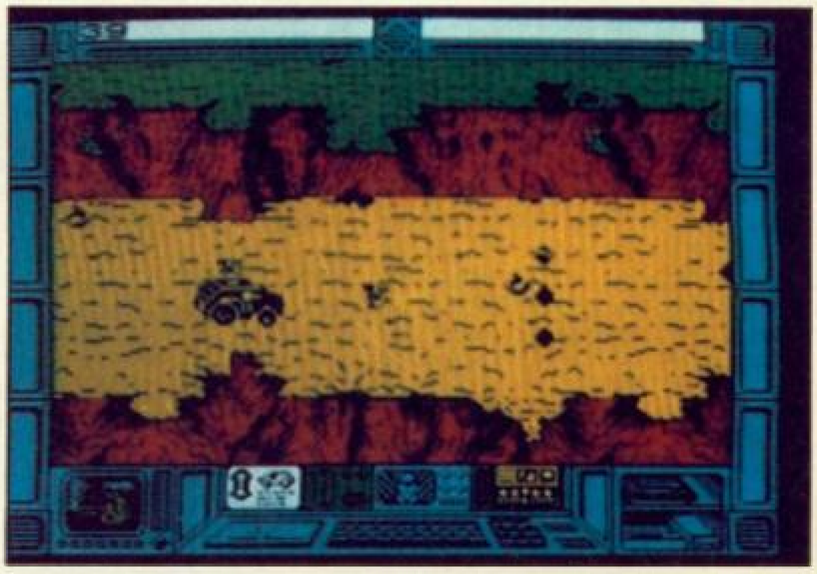
BREAKTHRU
US Gold
£8.99

The first of the US Gold coin-op conversions is Data East's Breakthru starring an acrobatic van!

This van is described as the world's most sophisticated armed vehicle that must drive 400 miles across hostile territory to retrieve your county's revolutionary new fighter and restore world peace.

Ahead of you lies four stages in which you must run the gauntlet of flame throwers, helicopters, tanks, jeeps, mines and enemy troops that can take out your sophisticated vehicle with a single shot!

The first stage is a charge through the mountains hurdling rockfalls in a single leap (Yes this van can jump!). Next you've got to cross a broken bridge



(more leaping) while battling with missile firing trucks, then across a prairie, through a city until finally you breakthru to the airfield when you can run for a plane and a final getaway.

A jumping, shooting, leaping arcade hit.



ORBIX THE TERRORBALL
Streetwise
£8.95

A bouncing laser firing ball called Orbix is the hero of the debut game from Domark's new arcade game label.

Orbix bounces and blasts his way through a four way scrolling landscape in search of the components of a lost spacecraft, rebuild it, find it's crew and let them escape!

The ship crashed on Horca, a planet infested with insectivores and droids that now gang up on you.

Their touch drains your energy but this can be replaced by chomping their remains once you've fried them with your laser.

As the game continues your quest is constantly interrupted by the need to replenish your energy which has been drained by almost constant attack.

Orbix is also supplied with a series of maps that are almost entirely unused since the on screen scanners guide you around the screen.

I found the controls irritating (rotate left/right and forward) and would prefer a more direct system particularly to steer through the landscape of factories, palm trees, towers and bubbling holes. Despite this, it's a good debut by a new label and a new programmer.

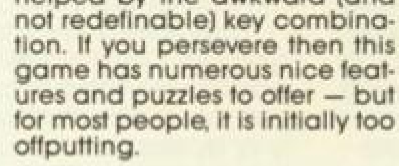


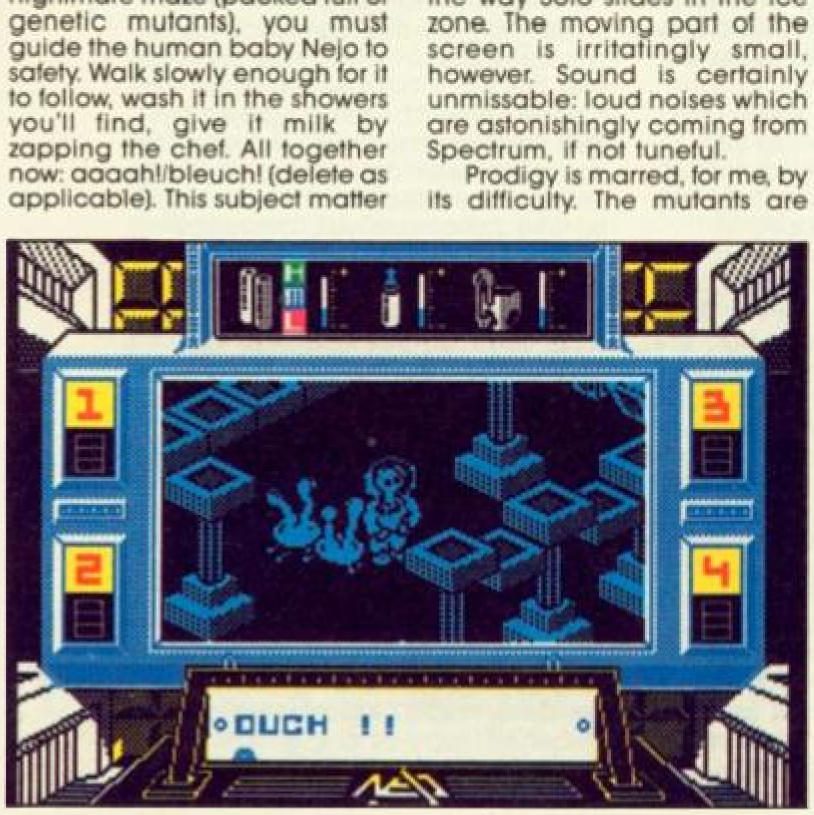
PRODIGY
Electric Dreams
£7.99

"Prodigy is a game which demands compassion" claims the packaging. Compassion? Yes, 'fraid so, not just shooting things. You are Solo, a synthetic man, and as well as escaping from the Machine Sorcerer's nightmare maze (packed full of genetic mutants), you must guide the human baby Nejo to safety. Walk slowly enough for it to follow, wash it in the showers you'll find, give it milk by zapping the chef. All together now: aaaah!/bleuch! (delete as applicable). This subject matter is one of several areas where Prodigy is rather original. It combines various gameplay elements to create a unique "feel".

The graphics are nicely defined and detailed in the 3D Knight Lore style; furthermore, they scroll rather than flick between locations, remarkably smoothly. There are four distinctive zones (fire, vegetation, technical and ice), each with its own graphics. A nice touch is the way Solo slides in the ice zone. The moving part of the screen is irritatingly small, however. Sound is certainly unmissable: loud noises which are astonishingly coming from Spectrum, if not tuneful.

Prodigy is marred, for me, by its difficulty. The mutants are extremely hard to avoid once near you, and instead of diminishing your energy, collision sends you back to the nearest teleporter or the start, whichever is nearest. This is unbelievably frustrating, and matters are not helped by the awkward (and not redefinable) key combination. If you persevere then this game has numerous nice features and puzzles to offer — but for most people, it is initially too offputting.







W.A.R.
Martech
£7.95

Fast action games seem to be making a comeback led, undoubtedly by the long awaited conversion of Hewson's Uridium.

In it's original C64 format, W.A.R. was little more than an Uridium clone but it has changed dramatically during it's conversion.

The action takes place in a tiny window with the rest of the screen there to add atmosphere.

In this window your laser firing fighter must blast the assembled aliens and destroy a scrolling spaceship background.

As you clear these levels you advance along the tubular space station illustrated on the game's cover.

W.A.R. is saved from the obscurity of being yet another shoot 'em up by the inclusion of a separate advanced version on the other side of the tape and the ability to trade hard earned points to improve your ship.

The Captains version, as it's known, is fought over a giant circuit board that is protected by incredibly violent alien ships. In this advanced test your standard ship won't last very long.

By trading points you can actually add to your fire power and buy extra lasers, more weapons and even a bonus ship. This might just save you. It saved the game.





TERRA COGNITA
Code Masters
£1.99

This is one of the first batch of games from a new budget software house — Code Masters.

The company was created by the Darlings who made their name through budget kings Mastertronic and are now going it alone.

This game is actually written by Non Terraqueous author Stephen Curtis and is a fast action shoot-em-up featuring great graphics, addictive action and amazing sound.

The plot is tenuous, involving a robot's revenge on a team of mining engineers who must escape a hostile landscape while being attacked by hordes of aliens.

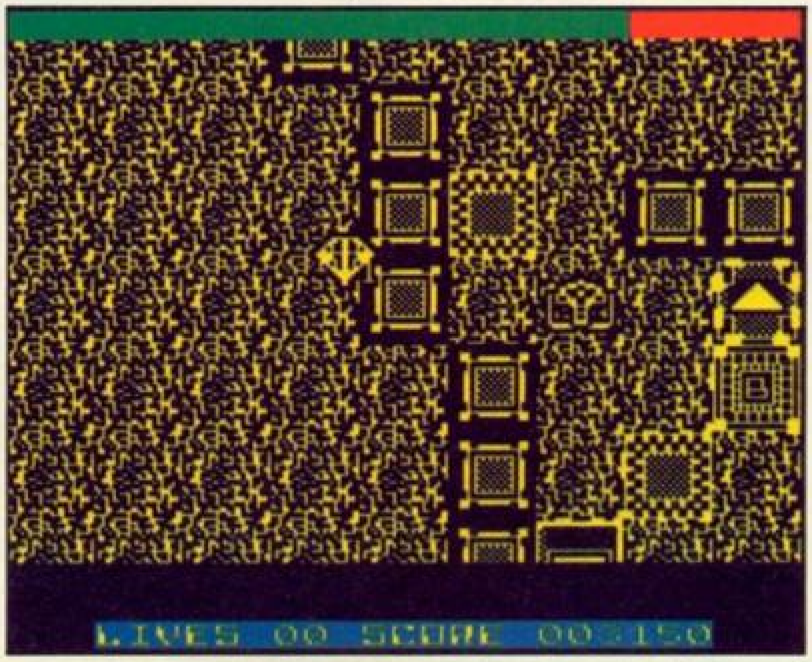
During the game you must fly your ship through a hundred screens of alien landscapes avoiding walls that will destroy you.

Built into the landscape are squares or zones that have a varying effect on you. Some speed you up or slow you down, give you extra lives and top up your fuel tanks but some are time shifts that return you to screen one. Irritating, particularly if you've reached screen 98!

The action is fast and furious with the sound of the laser sounding like a gunshot from a spaghetti western.

My only complaint is that you must select your joystick or keyboard option before every game. Pressing the fire button returns you to keyboard mode!





180
Mastertronic — MAD range
£2.99

Up to the oche comes yet another darts game. Through to the quarter finals of the championship knockout, you go into the draw with such giants of the arrows as Delboy Des, Sureshot Sidney (a real gas), Belly Bill and Limp Wrist Larry.

The game is 501 straight in, double out, played over the best of three legs although the computer sometimes cuts this to one leg for no obvious reason that I can see save that it is always the computer that has happened to win that particular leg. You always get to throw first, a decided tactical advantage and one that is crucial in the final against Jammy Jim who throws nine dart finishes as regular as clockwork (seven treble twenties, treble nineteen and double twelve).

When it is your turn to throw, a large hand appears in front of the board, shaking so much, you can see why darts players have to keep knocking back the pints. Movement round the screen is diagonal to put you even further off your aim.

Keyboard operation is decidedly easier than using a joystick. A scoreboard on the left of the board chalks up your scores as each arrow thuds (hopefully) into the board. Should you achieve a maximum 180, the computer greets you a rousing rendition of 'one hundred and eighty' although speech synthesis being what it is on the Spectrum, it comes out as more of a hissed 'nuh-nuh-nuh-neh-neh'.

Then it's time to sit back, slurp your beer and light another fag as your opponent has his three shots. The scene switches to a sideways view of the pub as Mega Mick or whoever takes his go. The blurb says to watch out for animated action in the background but all this seems to consist of is the barmaid sliding a pint along the bar to a customer and a small dog relieving itself against a chair leg.

Other options in the game include a practice facility in a version of round the clock and the chance to play against a fellow human should the pubs be shut. 180 is not one of Mastertronic's better offerings and I found it slightly surprising that they decided to bring it out on their more expensive MAD range. Like most of my darts, 180 is way off target.





KING'S KEEP
Firebird
£1.99



The country is ruled by a tyrant. The peasants know it. The merchants know it. What makes the fact that you know it even harder to bear is that King Harold the Heartless, as he has become known, is your father. Suspecting that your loyalties lie with the people rather than him, he has had you imprisoned in the depths of his castle. Can you escape and help to overthrow your father the despot or will you suffer his domination like the rest of the populace?

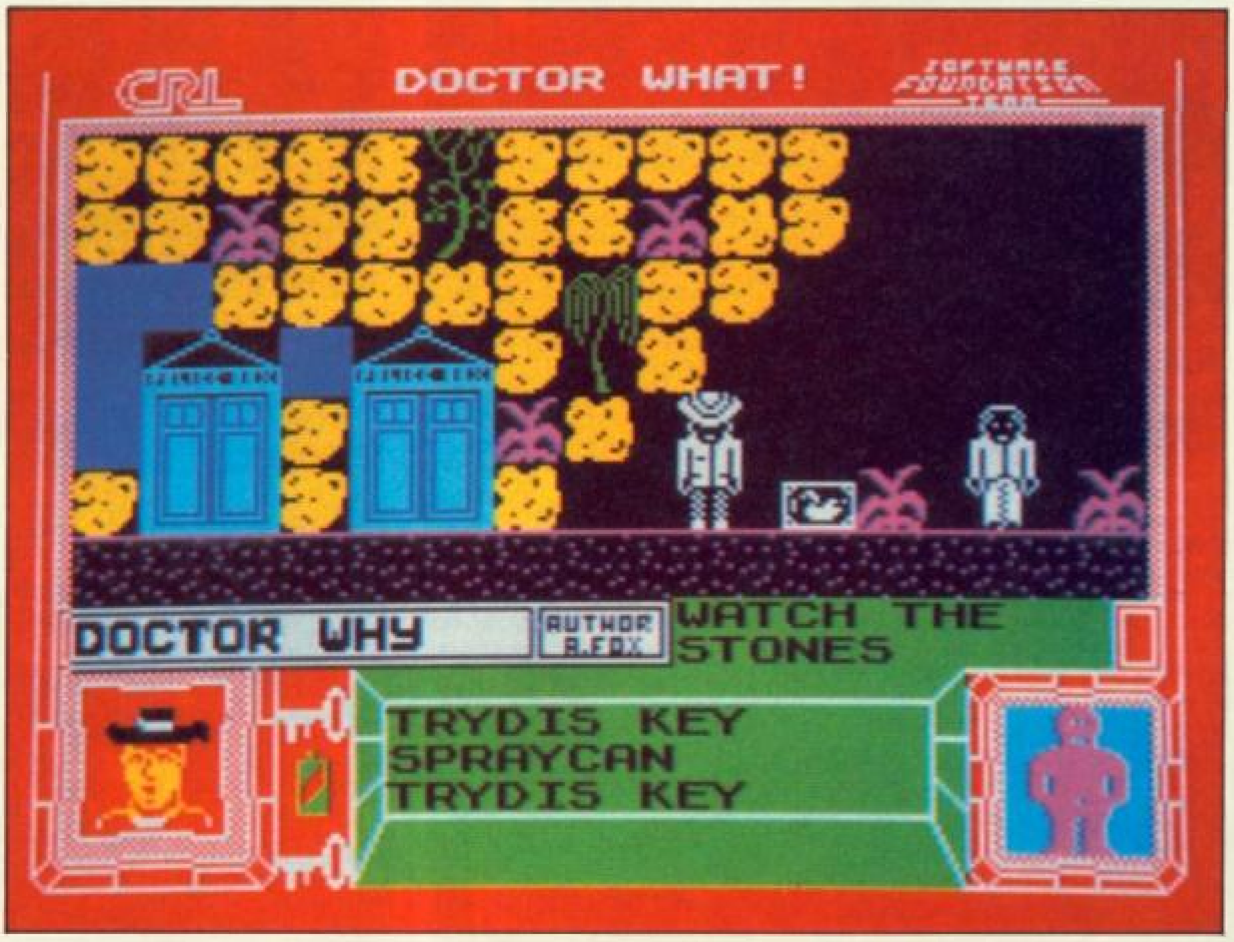
Like many arcade adventures being released at the moment, King's Keep has heavy platform game overtones. Dotted around the castle are various bits of furniture and brick wall that can, by means of a prodigious leap, be reached thus allowing you access to other parts of the castle. There is also a fair amount of objects to be collected on your travels and people to talk to.

Pressing the 'S' key brings up a list of additional commands available to you. These range from talking to someone, giving them something or manipulating a previously acquired item. For example, the crazy old man has lost his necklace. As he is blocking the entrance to another part of the castle, it would seem prudent to try and find it in the hope that he will let you pass. Certain areas of the castle are marked with an 'F'. This means that you are forbidden to enter them until a certain task has been accomplished. Refusal to obey results in an instant game over message.

The game, if nothing too original or stunning to look at is quite playable, and if you enjoy the collect-the-objects-and-solve-the-puzzles type of game, then King's Keep is not bad value for money.



DR. WHAT
CRL
£7.95



After a heavy intergalactic party where the supplies of Neuro Cardial cocktails have taken a bit of a battering, the four Doctors, What, When, Why and Where are all suffering from various degrees of a hangover. What is even worse is that they all lost in four different places deep within the space time continuum. Can they sort things out using their Trydis? This is beginning to sound like a spoof on a well known TV character. 'Who?' you ask. 'Correct', everyone replies.

The four Doctors have got to try and reach the Jelly Baby of Infinite Wisdom and Ultimate Knowledge who is stuck away in the Tower of Darabur somewhere. The first problem is to try and get the Doctors reassembled in one spot and quite tricky this proves to be. Why is in What's laboratory and is the only Doctor with access to a key, to allow him to enter and use the Trydis. Each Doctor can carry and use up to three articles at any given time. There is a row of buttons in each Trydis. Pressing a button teleports you to a specific location.

Each Doctor has only a limited amount of strength, depicted by a jelly baby being nibbled away in the bottom right hand corner of the screen. Collisions with the nasties, including a Dalek, are the major source of energy loss. Each of the Doctors can be summoned as and when you want them. But be warned. If one of the Doctors dies, your quest is effectively over so abort that particular game and start again.

As a game, Doctor What raises several questions; WHAT is the point of all this? WHERE is their dictionary so that they can learn how to spell 'transcendental' properly? WHEN will CRL realise that they can't get away with releasing poor quality, full price games? And WHY didn't they consign this load of rubbish to that great Jelly Baby in the Sky a long time ago?



BIG

SUBSCRIPTION SAVINGS FOR YOU!

Take advantage of these fantastic money saving subscription offers to these magazines. Hurry, this amazing offer is for a limited period only.

	UNITED KINGDOM			OVERSEAS (Accelerated surface post)		
	Normal Price	*Sale Price*	*Please Tick*	*Normal Price*	*Sale Price*	*Please Tick*
A&B Computing	~~£21.80~~	**£18.00**	☐	~~£26.00~~	**£20.80**	☐
Computer Gamer	~~£16.00~~	**£13.00**	☐	~~£21.50~~	**£17.20**	☐
Your Commodore	~~£16.00~~	**£13.00**	☐	~~£21.50~~	**£17.20**	☐
ZX Computing Monthly	~~£15.00~~	**£12.00**	☐	~~£18.00~~	**£14.40**	☐
Citizens' Band	~~£16.90~~	**£13.52**	☐	~~£22.00~~	**£17.60**	☐
Ham Radio Today	~~£17.30~~	**£14.40**	☐	~~£21.00~~	**£16.80**	☐
Electronics Digest	~~£11.30~~	**£ 8.30**	☐	~~£14.00~~	**£11.20**	☐
Electronics Today International	~~£18.10~~	**£14.40**	☐	~~£22.50~~	**£18.00**	☐
Video Today	~~£16.90~~	**£13.52**	☐	~~£21.00~~	**£16.80**	☐
Which Video?	~~£16.90~~	**£13.52**	☐	~~£21.00~~	**£16.80**	☐
Photography	~~£16.00~~	**£12.00**	☐	~~£21.50~~	**£16.00**	☐
Photoplay	~~£17.90~~	**£14.32**	☐	~~£21.50~~	**£17.20**	☐
Clocks	~~£30.80~~	**£24.64**	☐	~~£35.00~~	**£28.00**	☐
Woodworker	~~£16.90~~	**£13.52**	☐	~~£21.00~~	**£16.80**	☐
Popular Crafts	~~£17.90~~	**£14.32**	☐	~~£21.50~~	**£17.20**	☐
Winemaker & Brewer	~~£13.70~~	**£10.96**	☐	~~£17.00~~	**£13.60**	☐
Aeromodeller	~~£25.10~~	**£20.08**	☐	~~£29.00~~	**£23.20**	☐
Military Modelling	~~£16.90~~	**£13.00**	☐	~~£21.00~~	**£16.80**	☐
Model Boats	~~£16.10~~	**£13.00**	☐	~~£20.00~~	**£16.00**	☐
Radio Control Model Cars	~~£19.10~~	**£14.00**	☐	~~£21.50~~	**£17.20**	☐
Model Engineer	~~£27.40~~	**£22.00**	☐	~~£32.50~~	**£26.00**	☐
Radio Control Boat Modeller	~~£ 8.50~~	**£ 7.50**	☐	~~£11.50~~	**£ 9.20**	☐
R C M & E	~~£15.80~~	**£12.00**	☐	~~£20.00~~	**£16.00**	☐
Radio Control Scale Aircraft Quarterly	~~£ 9.70~~	**£ 8.00**	☐	~~£11.50~~	**£ 9.20**	☐
Radio Modeller	~~£16.10~~	**£13.00**	☐	~~£20.00~~	**£16.00**	☐
Sea Classic International	~~£10.30~~	**£ 9.00**	☐	~~£12.50~~	**£10.00**	☐
Scale Models International	~~£18.00~~	**£13.00**	☐	~~£20.00~~	**£16.00**	☐
Your Model Railway	~~£16.00~~	**£12.00**	☐	~~£25.00~~	**£20.00**	☐

(Offer ends 30th April 1987)

Please commence my subscription(s) with the .... issue.

I enclose my cheque/postal order for £............ made payable to Argus Specialist Publications Ltd.

or debit £............ from my Access/Barclaycard No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

valid from .... to .... Signature ....

Name ....

Address ....

....

Send this form with your remittance to: *Subscriptions Savings Offer (S.087)*
INFONET LTD., Times House, 179 The Marlowes, Hemel Hempstead, Herts. HP1 1BB.

SPECWORD 48/128

Part two of a menu driven wordprocessor for both 48 and 128 Spectrums by Stuart Nichols.

Listing 1

```
   1 REM ***************
   2 REM ** HEXLOADER **
   3 REM ***************
   4 REM
   5 PAPER 7: INK 0: CLS : PRINT AT 7,7;"1. HEXLOADER": PRINT AT 9,7;"2. HEXDUMP
";AT 15,2;"Please select option 1 or 2"
   6 LET a$=INKEY$: IF a$<>"1" AND a$<>"2" THEN  GO TO 6
   7 IF a$="2" THEN  GO TO 1000
  10 DEF FN a()=INT (y/16)
  20 DEF FN b()=y-FN a()*16
  30 DEF FN c()=INT (x/256)
  40 DEF FN d()=x-FN c()*256
  50 DEF FN e()=INT (t/256)
  60 DEF FN f()=t-FN e()*256
  70 DEF FN g(a$,b)=CODE a$(b)-48-7*(CODE a$(b)>57)
  80 DEF FN h(a$)=16*FN g(a$,1)+FN g(a$,2)
  90 DEF FN c$()=CHR$ (FN a()+48+7*(FN a()>9))+CHR$ (FN b()+48+7*(FN b()>9))
 100 PAPER 7: INK 0: CLS
 110 LET r$="0"
 120 PRINT AT 11,7; FLASH 1;"REM SET CAPS LOCK";AT 13,3;"HAVE YOU CLEARED MEMORY
 ?"
 130 INPUT "Start address (DECIMAL) ";adr
 135 LET beg=adr
 140 PAPER 7: INK 0: CLS
 150 CLS : LET x=adr
 160 PRINT x
 180 INPUT "8 bytes + CHK. ";b$
 190 IF b$="" THEN  GO TO 180
 200 IF b$="ZZ" THEN  GO TO 450
 210 LET st=adr
 220 LET le=LEN b$: IF INT (le/2)*2<>le THEN  PRINT "Odd no. of chrs. ";: GO TO
410
 230 IF le=18 THEN  GO TO 250
 240 IF b$(le-1 TO )<>"ZZ" THEN  PRINT "No 'ZZ' marker ";: GO TO 410
 250 FOR a=1 TO le-2: IF b$(a)<"0" OR b$(a)>"F" THEN  PRINT "Invalid chr. ";: GO
 TO 410
 260 IF b$(a)<"A" AND b$(a)>"9" THEN  PRINT "Invalid chr. ";: GO TO 410
 270 NEXT a
 280 LET x=0
 290 FOR k=1 TO 8
 300 LET a$=b$( TO 2): LET b$=b$(3 TO )
 310 IF b$="ZZ" THEN  LET k=9: LET r$=b$: LET b$=a$: GO TO 360
 320 PRINT a$;" ";
 330 LET p=FN h(a$)
 340 LET x= x+p
 350 POKE adr,p: LET adr=adr+1
 360 NEXT k
 370 LET y=FN d(): PRINT "= ";FN c$();"/";b$
 380 LET a$=b$
 390 IF y=FN h(a$) THEN  GO TO 430
 400 PRINT "Checksum ";
 410 PRINT "error -re enter": BEEP .25,5: BEEP .5,2: LET adr=st
 420 GO TO 180
 430 BEEP .1,20
 440 IF r$<>"ZZ" THEN  GO TO 150
 450 PRINT "PROGRAM TERMINATED"
 460 PRINT "Address start = ";beg
 470 PRINT "Address   end = ";adr-1
 480 PRINT "Length of code = ";adr-beg
 490 STOP

 900 REM
 910 REM
 920 REM ***********************
 930 REM ** HEXDUMP+CHECKSUM **
 940 REM ***********************
 950 REM
1000 PAPER 7: INK 0: CLS : LET p=0
1010 INPUT "(S)creen OR (P)rinter ? ";q$
1020 IF q$="P" OR q$="p" THEN  LET p=1: GO TO 1040
1030 IF q$<>"S" AND q$<>"s" THEN  BEEP .5,2: GO TO 1010
1040 INPUT "From address (DECIMAL) ";st
1050 INPUT "To address (DECIMAL) ";end
1060 PAPER 7: INK 0: CLS
1070 FOR x=st TO end STEP 8
1080 LET y=FN c(): IF p THEN  LPRINT x;" ";: GO TO 1100
1085 PRINT x;" "
1100 LET t=0
1110 FOR z=0 TO 7
1120 IF x+z>end THEN  LET z=8: GO TO 1150
1130 LET y=PEEK (x+z): LET t=t+y
1140 IF p THEN  LPRINT FN c$();" ";: GO TO 1150
1145 PRINT FN c$();" ";
1150 NEXT z
1155 LET y=FN f()
1160 IF p THEN  LPRINT ;"= ";FN c$();: GO TO 1170
1165 PRINT "= ";FN c$();
1170 IF x+z>end THEN  GO TO 1210
1180 IF p THEN  LPRINT : GO TO 1200
1190 PRINT
1200 NEXT x
1210 IF p THEN  LPRINT "ZZ": STOP
1220 PRINT "ZZ"
```

In last month's article you received the Basic program. Now it's time to begin the machine code section.

Using the HEXLOADER/HEXDUMP dual purpose program enter the machine code as HEXDUMP LIST 2.

Remember to reset ramtop to start address-1 before running this hexloader (ie after clear 33535 as a direct command for list 2).

The last byte in each line is the checksum byte, this being the sum of the previous 8 bytes modulo 256. The code should be entered one LINE at a time as a block of 18 characters (8 bytes + checksum) WITH NO SPACES BETWEEN THE CHARACTERS. To end the Hexloader program enter ZZ after the checksum byte. This can be done after any line in the dump and a printout of the number of bytes entered will be given (allowing you to save the block 'so far') and the last address. you will then be able to continue entering code from where you left off.

The HEXLOADER program will error trap **any** incorrect line inputs and prompt you to re enter wrongly keyed in lines.

SAVE this block of code as "code1" CODE 33536,9407

Next month's article contains the final part of the Machine Code and the full operating manual for Specword.

Listing 2

```
ZX Spectrum HEXDUMP
SPECWORD 48/+128K LIST 2
33536   C3 07 8A C3 0A 8A C3 7F = ED
33544   8C C3 D4 8C C3 0A 8A C3 = C9
33552   87 84 C3 13 85 C3 21 85 = CF
33560   F3 21 00 80 11 01 80 01 = 27
33568   00 01 36 81 ED B0 21 81 = F7
33576   81 36 C3 11 3F 83 23 73 = E3
33584   23 72 3E 80 ED 47 ED 5E = D2
33592   21 6C 83 36 00 FB C9 F3 = FD
33600   E5 D5 C5 F5 3A 08 5C 4F = 61
33608   FD CB 01 6E 28 17 FD CB = 3E
33616   01 AE 21 A2 82 ED 5B 6C = A8
33624   83 7B FE 40 28 07 13 19 = 97
33632   ED 53 6C 83 71 F1 C1 D1 = 23
33640   E1 C3 38 00 00 00 F3 3E = 0D
33648   3F ED 47 ED 56 FB C9 11 = 8B
33656   20 C0 21 C0 83 01 31 00 = 76
33664   ED B0 21 9F 83 11 00 C0 = B1
33672   01 1F 00 ED B0 FD CB 30 = B5
33680   9E 21 91 82 54 5D 36 00 = B9
33688   13 01 1F 00 ED B0 C9 21 = BA
33696   C0 4F C0 3F C0 3F C0 50 = 1D
33704   C0 00 00 00 00 00 00 00 = C0
33712   F8 00 00 00 00 00 0F 07 = 0E
33720   05 00 00 00 00 00 00 00 = 05
33728   0D 0D 0D 0D 0D 0D 0D 0D = 68
33736   0D 0D 0D 0D 0D 0D 0D 0D = 68
33744   0D 0D FF 66 72 65 64 FF = B9
33752   FF FF FF FF FF FF 0D 0D = 14
33760   FF FF D4 E5 F8 F4 FF C5 = 67
33768   EE E4 FF FF 0D 0D 0D 0D = 04
33776   FF D4 0F FF FF FF F4 12 = E5
33784   FF FF FF D9 1B 2D 01 FF = 1E
33792   F9 1B 2D 00 FF C9 1B 45 = 69
```

```
33800  FF FF E9 1B 46 FF FF CF = 15
33808  1B 47 FF FF EF 1B 48 FF = B1
33816  FF D0 1B 53 00 FF F0 1B = 47
33824  54 FF FF C7 1B 53 01 FF = 87
33832  E7 1B 54 FF FF C8 1B 57 = 8E
33840  01 FF E8 1B 57 00 FF CA = 23
33848  1B 30 FF FF EA 1B 32 FF = 7F
33856  FF CB 1B 31 FF FF EB 1B = 1A
33864  32 FF FF CC 1B 38 FF FF = 4D
33872  EC 1B 39 FF FF D6 0A FF = 1D
33880  FF FF F6 FF FF FF FF C2 = B2
33888  1B 51 8E FF E2 1B 51 50 = 97
33896  FF CE 0E FF FF FF EE 14 = DA
33904  FF FF FF CD 1B 55 01 FF = 3A
33912  ED 1B 55 00 FF D5 1B 34 = 80
33920  FF FF F5 1B 35 FF FF 2A = 6B
33928  4B 5C 3E 59 BE 28 25 CB = 14
33936  6E 20 09 23 5E 23 56 19 = A9
33944  23 18 EF CB 76 20 0B 23 = B9
33952  CB 7E 28 FB 11 06 00 19 = 9C
33960  18 E0 CB 7E 20 F6 11 13 = 83
33968  00 19 18 D6 23 4E 23 46 = E1
33976  22 F0 84 ED 43 F2 84 ED = 29
33984  5B 04 C0 13 2A F0 84 D5 = A5
33992  ED 4B F2 84 23 1A 13 BE = BC
34000  28 05 F6 20 BE 20 08 0B = 34
34008  78 B1 20 F0 D1 18 0A D1 = FD
34016  13 1A FE FF 20 DE C3 62 = 4D
34024  88 ED 53 04 C0 C3 0A 8A = E6
34032  C0 7E 01 00 11 FA 84 C3 = 91
34040  6F 8B 10 07 11 02 16 0B = 45
34048  06 20 52 49 47 48 54 20 = C4
34056  4D 41 52 47 49 4E 20 53 = 31
34064  45 54 20 21 3F C0 22 1F = 1A
34072  85 3E 12 32 ED 8C C9 3F = 88
34080  FF 3E 1E 32 ED 8C CD 0C = DF
34088  90 22 1F 85 ED 5B 08 C0 = 66
34096  A7 ED 52 30 0E 2A 1F 85 = F2
34104  01 00 00 28 7E FE 0D C8 = 7D
34112  03 18 F8 01 FF FF C9 CD = A8
34120  43 92 CD 6A 92 CD 48 8B = 3E
34128  2A 80 92 7C B5 CA 1C 8A = DD
34136  2A 08 C0 E5 ED 5B 04 C0 = E3
34144  A7 ED 52 22 13 93 E1 E5 = 74
34152  ED 5B 80 92 19 22 08 C0 = 5D
34160  EB E1 ED 4B 13 93 03 ED = 9A
34168  B8 CD 0C 90 23 ED 5B 04 = 90
34176  C0 ED 4B 80 92 ED B0 C3 = 6A
34184  19 8A AF 32 42 92 21 00 = 79
34192  01 22 40 92 2A 3C 92 11 = FE
34200  90 81 01 00 01 ED B0 21 = D1
34208  90 81 01 00 01 3E 19 ED = 57
34216  B1 20 05 3E 01 32 42 92 = 1B
34224  11 90 81 A7 ED 52 22 40 = 6A
34232  92 2A 3C 92 ED 5B 40 92 = A4
34240  19 44 4D 2A 04 C0 A7 ED = 2C
34248  42 C5 44 4D E1 ED 5B 3C = FD
34256  92 ED B0 21 90 81 ED 4B = 99
34264  40 92 ED B0 3A 42 92 A7 = 24
34272  28 B2 CD 0C 90 22 04 C0 = 29
34280  C3 10 8A F3 3A 6C 83 A7 = 20
34288  28 14 21 A2 82 23 7E ED = 0F
34296  4B 6C 83 0B ED 43 6C 83 = 64
34304  03 54 5D 23 ED B0 FB A7 = 16
34312  C8 FE 06 C0 3A 6A 5C EE = 7A
34320  08 32 6A 5C 3A 6D 91 EE = 26
34328  01 32 6D 91 76 CD 07 91 = 0C
34336  18 C9 21 60 5A ED 5B 0E = 12
34344  C0 19 3A 11 C0 77 C9 21 = 45
34352  60 5A ED 5B 0E C0 19 ED = D6
34360  4B 18 C0 71 C9 2A 0C C0 = 53
34368  23 22 0C C0 2A 04 C0 23 = 22
34376  22 04 C0 C9 CD 71 89 21 = 97
34384  0E C0 7E FE 1C 28 0B 34 = CD
34392  2A 04 C0 23 22 04 C0 C3 = BA
34400  8B 87 CD 3D 86 C3 83 87 = 6F
34408  2B 28 01 FF FF 3E 0D ED = 8D
34416  B9 23 23 C9 2A 0C C0 2B = E9
34424  22 0C C0 2A 04 C0 2B 22 = 29
34432  04 C0 C9 2A 04 C0 ED 5B = C3
34440  06 C0 A7 ED 52 E5 11 1C = BE
34448  00 ED 52 38 09 22 0C C0 = 6E
34456  ED 53 0E C0 E1 C9 E1 22 = BB
34464  0E C0 21 00 00 22 0C C0 = DD
34472  C9 CD D9 86 21 91 82 01 = 2A
34480  20 00 ED B1 28 20 21 91 = B8
34488  82 01 20 00 AF ED B1 20 = 10
34496  15 2B 3A 01 87 77 3E 01 = B8
34504  CD 9B 22 06 0A 76 10 FD = 1D
34512  3A 19 C0 CD 9B 22 C3 1C = 7C
34520  8A 2A 0E C0 ED 5B 0C C0 = 96
34528  19 7C A7 28 03 E1 18 EE = 4E
34536  7D 32 01 87 C9 CD D9 86 = 2C
34544  21 91 82 01 20 00 ED B1 = F3
34552  20 DC 2B 36 00 3E 02 18 = B5
34560  C7 00 00 CD D9 86 3C 32 = 61
34568  01 87 28 CA 01 20 00 21 = BC
34576  91 82 ED B1 CA 1C 87 3A = 58
34584  01 87 18 EA 2A 06 C0 01 = 78
34592  FF FF 3E 0D ED B1 2B 22 = 34
34600  04 C0 ED 5B 06 C0 A7 ED = 66
34608  52 ED 5B 01 87 E5 A7 ED = 9B
34616  52 30 0B E1 EB A7 ED 52 = 3F
34624  CD 21 8E C3 13 8A C1 7C = 19
34632  B5 CA 13 8A 2A 06 C0 19 = 25
34640  22 04 C0 C3 13 8A CD 91 = A4
34648  83 3E 04 C3 C8 86 ED 5B = 1E
34656  04 C0 2A 08 C0 4F 23 7C = A4
34664  B5 C8 79 A7 E5 ED 52 44 = 05
34672  4D E1 22 08 C0 54 5D 2B = F4
34680  ED B8 12 2A 02 C0 23 22 = E8
34688  02 C0 C9 11 20 40 2A 00 = 26
34696  C0 18 06 11 60 50 2A 06 = CF
34704  C0 D5 ED 4B 0C C0 78 B1 = C2
34712  28 08 3E 0D ED B1 0E 21 = 48
34720  28 3A 0E 20 7E 23 F5 06 = 2C
34728  08 E5 D5 D5 26 00 6F 29 = 55
34736  29 29 ED 5B 36 5C 19 D1 = 16
34744  7E 12 23 14 10 FA D1 E1 = 83
34752  1C F1 FE 0D 28 16 0D 20 = 83
34760  D8 CD 7C 88 D1 3E 20 83 = 5E
34768  5F 20 BE 7A C6 08 57 FE = DA
34776  58 C8 18 B5 79 3D 28 EC = B7
34784  0E 08 EB 47 E5 36 00 23 = 86
34792  10 FB E1 24 0D 20 F4 EB = 1C
34800  18 DA CD 5E 87 A7 C8 2A = 3D
34808  00 C0 CD 77 88 22 00 C0 = 6E
34816  21 00 00 22 0C C0 22 0E = 3F
34824  C0 2A 04 C0 23 22 04 C0 = B7
34832  22 06 C0 CD 91 89 CD 5B = F7
34840  89 C9 2A 04 C0 7E FE 0D = C9
34848  28 1F 23 22 04 C0 CD 71 = 8E
34856  89 3A 0E C0 FE 1C 20 0A = D5
34864  2A 0C C0 23 22 0C C0 C3 = CA
34872  83 87 3C 32 0E C0 C3 8B = 94
34880  87 23 7E 3C C8 CD 5B 88 = DC
34888  2A 04 C0 23 22 04 C0 CD = C4
34896  83 86 CD 91 89 CD 5B 89 = A1
34904  C3 83 87 2A 00 C0 CD 77 = FB
34912  88 22 00 C0 2A 06 C0 CD = 27
34920  77 88 22 06 C0 2A 02 C0 = D3
34928  CD 77 88 22 02 C0 C9 7E = F7
34936  23 FE 0D C8 3E 0D 01 FF = 41
34944  FF ED B1 C9 2A 04 C0 2B = 7F
34952  7E FE 0D 28 44 CD 95 88 = DF
34960  CD 81 89 18 25 E5 ED 5B = 41
34968  08 C0 EB A7 ED 52 44 4D = 2A
34976  D1 62 6B 23 ED B0 1B ED = 66
34984  53 08 C0 2A 04 C0 2B 22 = 56
34992  04 C0 2A 02 C0 2B 22 02 = FF
35000  C0 C9 3A 0E C0 A7 20 07 = 67
35008  3D 32 0E C0 C3 8B 87 2A = 3C
35016  0C C0 2B 22 0C C0 C3 83 = 2B
35024  87 2B 7E 3C C8 2A 00 C0 = 1E
35032  CD 68 86 22 00 C0 2A 06 = CD
35040  C0 CD 68 86 22 06 C0 2A = 8D
35048  04 C0 2B CD 95 88 CD 83 = 29
35056  86 CD 32 8B CD A1 89 C3 = CA
35064  83 87 6F 26 00 29 29 29 = 1A
35072  11 00 AF 19 EB 21 00 40 = 25
35080  09 1A 77 13 24 7C E6 07 = 3A
35088  20 F7 C9 3E AF 32 37 5C = 92
35096  18 0A 21 6A 5C CB 9E 21 = 93
35104  6D 91 36 24 3E 02 CD 01 = 66
35112  16 21 00 58 01 1F 00 11 = C0
35120  01 58 36 47 ED B0 CD 74 = B4
35128  9B CD C6 A6 18 00 00 00 = F4
35136  00 00 00 00 00 00 3A 19 = 53
35144  C0 CD 9B 22 CD 5B 89 21 = 1C
35152  6C 89 11 5B 91 01 05 00 = F8
35160  ED B0 C9 21 67 89 11 51 = D9
35168  91 01 05 00 ED B0 C9 30 = 2D
35176  30 30 30 31 30 30 30 30 = 81
35184  31 21 55 91 7E 3C FE 3A = 2A
35192  20 05 36 30 2B 18 F5 77 = 3A
35200  C9 21 55 91 7E 3D FE 2F = B8
35208  20 05 36 39 2B 18 F5 77 = 43
35216  C9 21 5F 91 7E 3C FE 3A = CC
35224  20 05 36 30 2B 18 F5 77 = 3A
35232  C9 21 5F 91 7E 3D FE 2F = C2
35240  20 05 36 39 2B 18 F5 77 = 43
35248  C9 2A 04 C0 2B 7E FE 0D = 6B
35256  28 1A 3A 0E C0 A7 20 09 = 1A
35264  CD 74 86 CD 81 89 C3 83 = E4
35272  87 3D 32 0E C0 22 04 C0 = AA
35280  CD 81 89 C9 2B 7E 3C C8 = 4D
35288  23 22 04 C0 CD EB 89 CD = 17
35296  83 86 CD A1 89 CD 32 8B = 8A
35304  C3 83 87 2A 00 C0 CD 68 = EC
35312  86 22 00 C0 2A 06 C0 CD = 25
35320  68 86 22 06 C0 2A 02 C0 = C2
35328  CD 68 86 22 02 C0 C9 CD = 35
35336  77 83 CD 13 89 CD 18 83 = CB
35344  CD 72 90 CD 83 86 CD 32 = A4
35352  8B CD 83 87 CD 22 86 CD = A4
35360  07 91 CD EB 85 A7 28 FA = 9E
35368  CB 7F 20 F6 CD 2F 86 FE = E0
35376  0A 20 05 CD DF 8B 18 E4 = 62
35384  FE 0B 20 05 CD 4D 8C 18 = EC
35392  DB 21 78 8C 36 00 FE 07 = 3B
35400  20 0C 01 00 00 CD 6E 83 = EB
35408  21 6C 83 36 00 C9 FE 0F = 1C
35416  CA 4B 8E 10 11 2A 04 C0 = BA
35424  01 FF FF 3E 0D ED B1 28 = 13
35432  22 04 C0 C3 EC 8D FE 04 = 24
35440  38 FC CA 83 87 FE 05 CA = 55
35448  9D 93 FE 20 30 5C FE 0D = E5
35456  20 35 21 10 C0 CB 46 20 = 77
35464  0F CD F2 87 CA 67 8B 2A = 3B
35472  12 C0 CD 21 8E C3 13 8A = AE
35480  2A 06 C0 CD 77 88 7E 3C = 76
35488  3E 0D 20 E5 CD 5B 88 2A = 32
35496  06 C0 CD 77 88 28 22 04 = E3
35504  C0 CD 91 89 C3 EC 8D FE = E1
35512  0E CA 0F 8D FE 0C 20 06 = A4
35520  CD 84 88 C3 1C 8A FE 08 = 48
35528  20 06 CD B1 89 C3 1C 8A = 96
35536  FE 09 20 06 CD 1A 88 C3 = 5F
35544  1C 8A 21 10 C0 CB 46 20 = C8
35552  25 F5 2A 0C C0 ED 5B 0E = 66
35560  C0 19 ED 5B 14 C0 7A B3 = 22
35568  28 06 A7 ED 52 CA 16 8F = 83
35576  F1 CD 5E 87 A7 CA 67 8B = 06
35584  CD 4C 86 C3 1C 8A F5 2A = 27
35592  0C C0 ED 5B 0E C0 19 ED = E8
35600  5B 14 C0 7A B3 28 09 A7 = 34
35608  ED 52 20 04 E1 C3 F4 84 = 7F
35616  F1 2A 04 C0 4F 7E FE 0D = B7
35624  79 28 B6 71 CD 4C 86 C3 = 2A
35632  1C 8A CD 5B 89 2A 0C C0 = 4D
35640  ED 5B 0E C0 19 7C B5 C8 = 28
35648  E5 CD 71 89 E1 2B 18 F5 = C5
35656  2A 04 C0 ED 5B 3C 92 A7 = AB
35664  ED 52 D8 2A 04 C0 ED 5B = 4D
35672  3E 92 A7 ED 52 D0 E1 C3 = 2A
35680  1C 8A 11 C6 8B 18 08 11 = 39
35688  94 8B 18 03 11 AD 8B D5 = 58
35696  3E 02 CD 01 16 D1 01 19 = 0F
35704  00 CD 3C 20 06 64 76 10 = 19
35712  FD 21 3F C0 01 C0 3F 3E = 5B
35720  01 ED B1 C2 0A 8A 2B 36 = 56
35728  20 C3 0A 8A 10 07 11 02 = A1
35736  16 0B 06 20 20 54 45 58 = 58
35744  54 20 46 49 4C 45 20 46 = FA
35752  55 4C 4C 20 20 10 07 11 = 55
35760  02 16 0B 06 20 4E 4F 20 = 06
35768  52 4F 4F 4D 20 46 4F 52 = 44
35776  20 43 4F 50 59 20 10 07 = 92
35784  11 02 16 0B 06 20 57 4F = 00
35792  52 44 20 4E 4F 54 20 49 = 10
35800  4E 20 54 45 58 54 20 2A = FD
35808  06 C0 CD 77 88 7E 3C C8 = 14
35816  CD 91 89 3A 78 8C A7 20 = EC
35824  14 2A 0C C0 22 7D 8C ED = 22
35832  5B 0E C0 19 23 22 79 8C = 8C
35840  3E 01 32 78 8C CD 5B 88 = 25
35848  2A 06 C0 ED 4B 79 8C 78 = A5
35856  B1 28 05 3E 0D ED B1 2B = F2
35864  22 04 C0 ED 5B 06 C0 A7 = 9B
35872  ED 52 22 7B 8C ED 5B 7D = 2D
35880  8C ED 52 30 05 CD 83 86 = D6
35888  18 07 ED 53 0C C0 22 0E = 5B
35896  C0 CD 32 8B C3 83 87 FE = 15
35904  77 C2 1C 8A 00 00 00 01 = E0
35912  03 00 C3 4D 8A 2A 06 C0 = 8D
35920  CD 68 86 7E 3C C8 CD A1 = AB
35928  89 3A 78 8C A7 20 14 2A = CC
35936  0C C0 22 7D 8C ED 5B 0E = 4D
35944  C0 19 23 22 79 8C 3E 01 = 62
35952  32 78 8C CD EB 89 18 90 = 1F
35960  00 14 00 00 00 00 00 2A = 3E
35968  4F 5C 11 0F 00 19 11 8D = 82
35976  8C 73 23 72 C9 F5 3A FF = 8B
35984  BF CB 47 CC C0 8C F1 01 = DB
35992  BF E3 C5 05 1E 0E F5 CD = 5A
36000  54 1F D2 00 0D ED 50 CB = 5A
36008  3A 38 F4 F1 F3 05 05 ED = 41
36016  79 C1 ED 59 1C ED 59 FB = DD
36024  FE 0D C0 3E 0A 18 D8 00 = 03
36032  01 BF E3 3E 81 ED 79 3E = 06
36040  0F ED 79 3A FF BF CB C7 = FF
36048  32 FF BF C9 3E 03 CD 01 = C8
36056  16 2A 1F 85 7E FE FF 28 = 87
36064  2A FE C0 D2 EE 90 FE 18 = 4E
36072  28 16 FE 19 28 1E FE 60 = F9
36080  20 02 3E 81 FE 7F 20 02 = 80
36088  3E 60 D7 7E FE 0D 20 03 = 29
36096  23 18 D9 23 22 1F 85 01 = FE
36104  01 00 C9 01 00 00 C9 21 = B5
36112  67 91 36 25 CD 07 91 CD = 85
36120  EB 85 A7 28 FA 21 67 91 = 52
36128  36 24 FE 08 CA 5D 8A FE = 0F
36136  09 CA 85 8F FE 0B CA CA = 84
36144  90 FE 0A CA DD 90 FE 30 = 05
36152  CA F2 8F FE 39 CA FD 8F = D8
36160  FE 30 CA 69 90 FE 35 CA = EE
36168  47 85 FE 36 CA 82 92 FE = DC
36176  37 CA AE 91 FE 05 CA 9E = AB
36184  93 FE 31 CA A9 86 FE 32 = EB
36192  CA ED 86 FE 33 CA 56 87 = 15
36200  CB EF 21 7C 8D 01 08 00 = ED
36208  ED B1 20 18 11 07 00 19 = 07
36216  7E C3 DA 8A 79 75 70 61 = 64
36224  73 64 66 67 5B 5D 7F 7E = 59
36232  7C 5C 7B 7D FE 6F 20 12 = 6F
36240  21 10 C0 CB C6 21 11 C0 = 74
36248  CB CE 21 76 91 36 24 C3 = DE
36256  1C 8A FE 69 20 12 21 10 = 70
36264  C0 CB 86 21 11 C0 CB 8E = 5C
36272  21 76 91 36 25 C3 1C 8A = EC
36280  FE 71 20 29 CD 9D 91 2A = DD
36288  0C C0 ED 5B 0E C0 19 22 = 1D
36296  12 C0 ED 5B 14 C0 7A B3 = 1B
36304  CA 1C 8A A7 ED 52 DA 1C = 4C
36312  8A 21 00 00 22 12 C0 CD = 6C
36320  8D 91 C3 1C 8A FE 72 28 = 1F
36328  F0 C3 E2 8E ED 5B 06 C0 = 31
36336  2A 04 C0 A7 ED 52 E5 ED = A6
36344  5B 12 C0 A7 ED 52 30 0E = 51
36352  E1 ED 5B 12 C0 EB A7 ED = 7A
36360  52 CD 21 8E 18 10 D1 7C = 43
36368  B5 28 08 2A 06 C0 ED 5B = 20
36376  12 C0 19 22 04 C0 C3 13 = A7
36384  8A 7C B5 C8 E5 3E 20 CD = 93
36392  5E 87 A7 20 05 E1 E1 C3 = 36
36400  67 8B CD 39 8E E1 2B 18 = AA
36408  E8 21 0E C0 7E FE 1C CA = 39
36416  3D 86 34 2A 04 C0 23 22 = 2A
36424  04 C0 C9 21 63 91 36 25 = FD
36432  CD 07 91 CD EB 85 A7 28 = 71
36440  FA 21 63 91 36 24 FE 05 = 6C
36448  CA 9F 93 21 7A 8E 01 34 = 5A
36456  00 ED B1 C2 38 A7 11 33 = 83
36464  00 19 7E A7 CA 1C 8A C3 = 71
36472  DA 8A 41 61 42 62 43 63 = 50
36480  44 64 45 65 46 66 47 67 = AC
36488  48 68 49 69 4A 6A 4B 6B = CC
36496  4C 6C 4D 6D 4E 6E 4F 6F = EC
36504  50 70 51 71 52 72 53 73 = 0C
36512  54 74 55 75 56 76 57 77 = 2C
36520  58 78 59 79 5A 7A 96 96 = A2
36528  C2 E2 9A 9A 97 97 95 95 = 30
36536  9C 9C C7 E7 C8 E8 C9 E9 = 48
36544  CA EA CB EB CC EC CD ED = DC
36552  CE EE CF EF D0 F0 86 86 = 46
36560  9D 9D 9F 9F D4 F4 D5 F5 = 0A
36568  D6 F6 98 98 9E 9E D9 F9 = 0A
36576  99 99 FE 65 20 29 CD A9 = 54
36584  91 2A 0C C0 ED 5B 0E C0 = 9D
36592  19 22 14 C0 ED 5B 12 C0 = 29
36600  A7 ED 52 DA 03 8F 7C B5 = 83
36608  C2 1C 8A 21 00 00 22 14 = BF
36616  C0 CD 98 91 C3 1C 8A FE = 1D
36624  74 28 F0 C3 C5 8F F1 FE = 92
36632  20 20 05 3E 0D C3 89 8A = 66
36640  32 6C 8F 3E 01 CD 5E 87 = 1E
36648  A7 CA 67 8B 21 16 C0 CB = 25
36656  46 28 15 2A 14 C0 ED 5B = C9
36664  12 C0 A7 ED 52 44 4D 2A = 73
36672  04 C0 3E 20 ED B9 28 25 = 15
36680  3E 0D CD F2 87 A7 CA 67 = 69
36688  8B 2A 12 C0 CD 21 8E 2A = 2D
36696  04 C0 7E 3D 28 03 23 18 = E5
36704  F9 3A 6C 8F 77 23 22 04 = EE
36712  C0 C3 13 8A 7A 23 36 0D = 00
36720  23 22 04 C0 22 06 C0 2A = 1B
36728  00 C0 CD 77 88 22 00 C0 = 6E
36736  CD 91 89 18 CC 2A 06 C0 = BB
36744  01 FF FF 3E 0D ED B1 28 = 13
36752  22 04 C0 ED 5B 06 C0 A7 = 9B
36760  ED 52 ED 5B 14 C0 7A B3 = 88
36768  28 20 E5 A7 ED 52 30 0A = 4D
36776  E1 EB A7 ED 52 CD 21 8E = 2E
36784  18 10 D1 7C B5 28 08 2A = 87
36792  06 C0 ED 5B 14 C0 19 22 = 1D
36800  04 C0 C3 13 8A FE 6B 20 = AD
36808  12 21 71 91 36 25 21 16 = C7
36816  C0 CB C6 21 11 C0 CB A6 = B4
36824  C3 1C 8A FE 6A C2 15 93 = 3B
36832  21 71 91 36 24 21 16 C0 = 74
36840  CB 86 21 11 C0 CB E6 C3 = B7
```

SPECTRUM PROGRAMMING

```
36848  1C 8A CD 1D 90 CD 83 87 = F7
36856  3E 18 C3 DA 8A CD 18 90 = F2
36864  CD 83 87 3E 19 C3 DA 8A = 55
36872  3E 19 18 02 3E 18 21 3F = 27
36880  C0 01 C1 3F ED B1 2B C9 = 53
36888  CD 08 90 18 03 CD 0C 90 = E9
36896  ED 5B 08 C0 A7 E5 ED 52 = DB
36904  E1 D0 A7 ED 5B 06 C0 E5 = 4B
36912  ED 52 30 05 1B ED 53 06 = D5
36920  C0 E1 E5 ED 5B 04 C0 A7 = 39
36928  ED 52 30 05 1B ED 53 04 = D3
36936  C0 D1 2A 08 C0 A7 ED 52 = 69
36944  44 4D 62 6B 23 ED B0 1B = 39
36952  ED 53 08 C0 CD 72 90 CD = A4
36960  83 86 CD 32 8B CD 83 87 = 6A
36968  C9 CD 1D 90 CD 18 90 C3 = 7B
36976  19 8A CD 4F 89 2A 04 C0 = 36
36984  3E 0D 2B BE 20 FC 23 22 = 95
36992  06 C0 11 3F C0 A7 ED 52 = BC
37000  28 19 44 4D EB 11 00 00 = CE
37008  3E 0D BE 20 01 13 23 0B = 6B
37016  78 B1 20 F4 ED 53 0A C0 = 47
37024  CD B5 90 2A 06 C0 06 13 = 1B
37032  3E 0D 2B BE 20 FC 10 FA = 5A
37040  23 22 08 C0 C9 7A B3 C8 = C3
37048  21 5F 91 7E 3C FE 3A 20 = 23
37056  05 36 30 2B 18 F5 77 1B = 35
37064  18 EB 2A 04 C0 01 00 40 = 32
37072  3E FF ED B9 23 23 23 22 = 6E
37080  04 C0 C3 10 8A 21 3F C0 = 41
37088  01 00 40 3E FF ED B1 2B = 47
37096  2B 22 04 C0 18 EC E5 21 = 1B
37104  F1 83 01 96 00 ED B1 20 = C9
37112  09 7E FE FF 28 04 D7 23 = AA
37120  18 F7 E1 23 C3 DC 8C 21 = 5F
37128  20 00 11 00 40 01 4D 91 = 50
37136  C5 E5 D5 0A 26 00 6F 29 = 47
37144  29 29 11 00 B6 19 E5 03 = 1A
37152  0A 26 00 6F 29 29 29 19 = 33
37160  EB C1 E1 E5 0A 77 1A ED = FA
37168  6F 13 03 24 7C E6 07 20 = 32
37176  F3 D1 13 CB 42 28 04 3E = 4E
37184  07 82 57 E1 C1 03 03 2B = B3
37192  7C B5 20 C4 C9 43 4F 4C = BC
37200  3A 30 30 30 31 32 2A 4C = A3
37208  49 4E 3A 30 30 30 30 34 = C5
37216  2A 47 4D 24 2A 45 4D 24 = C2
37224  2A 43 41 50 53 24 2A 57 = F6
37232  57 24 2A 49 4E 53 25 2A = DE
37240  4C 4D 41 52 3A 30 30 30 = F6
37248  30 30 2A 52 4D 41 52 3A = F6
37256  30 30 30 30 30 21 7D 91 = 1F
37264  06 05 36 30 23 10 FB C9 = 68
37272  21 88 91 18 F3 11 7D 91 = 64
37280  21 51 91 01 05 00 ED B0 = A6
37288  C9 11 88 91 18 F2 CD 0C = D6
37296  90 CD 32 92 D2 1C 8A 22 = BB
37304  3C 92 CD 08 90 CD 32 92 = C4
37312  D2 1C 8A 22 3E 92 ED 5B = B2
37320  3C 92 A7 ED 52 DA 1C 8A = 34
37328  7C A7 20 0D 85 3D 20 09 = 3B
37336  CD 1D 90 CD 18 90 C3 19 = CB
37344  8A ED 5B 3E 92 2A 08 C0 = 94
37352  A7 ED 52 44 4D 2A 3E 92 = 71
37360  23 ED 5B 3C 92 ED B0 1B = F1
37368  EB 22 08 C0 2A 04 C0 ED = B0
37376  5B 3C 92 A7 ED 52 38 27 = 6E
37384  2A 04 C0 ED 5B 3E 92 A7 = AD
37392  ED 52 30 05 2A 3C 92 18 = 84
37400  10 2A 04 C0 ED 5B 3E 92 = 16
37408  A7 ED 52 ED 5B 3C 92 19 = 15
37416  2B 22 04 C0 C3 10 8A C3 = 31
37424  19 8A ED 5B 08 C0 A7 E5 = 3F
37432  ED 52 E1 C9 00 00 00 00 = E9
37440  00 00 00 21 00 00 22 80 = C3
37448  92 CD 0C 90 CD 32 92 D0 = 5C
37456  22 3C 92 CD 08 90 CD 32 = 54
37464  92 D0 22 3E 92 A7 ED 5B = 43
37472  3C 92 ED 52 D8 2B 22 80 = B2
37480  92 C9 ED 5B 08 C0 21 FF = 8B
37488  FF A7 ED 52 ED 5B 80 92 = 3F
37496  A7 ED 52 D0 E1 C3 6C 8B = 51
37504  00 00 CD 43 92 2A 80 92 = DE
37512  7C B5 CA 1C 8A 23 23 22 = 09
37520  80 92 2A 04 C0 ED 5B 3C = 84
37528  92 A7 ED 52 CA 1C 8A 38 = 20
37536  10 2A 04 C0 ED 5B 3E 92 = 16
37544  A7 ED 52 DA 1C 8A C3 8A = B3
37552  85 AF 32 42 92 21 00 01 = 5C
37560  22 40 92 2A 3E 92 11 90 = 8F
37568  82 01 00 01 ED B8 21 90 = DA
37576  82 01 00 01 3E 18 ED B9 = 80
37584  20 05 3E 01 32 42 92 11 = 7B
37592  90 82 EB A7 ED 52 22 40 = 45
37600  92 2A 3E 92 E5 ED 5B 40 = F9
37608  92 A7 ED 52 E5 ED 4B 04 = 99
37616  C0 A7 ED 42 44 4D 03 E1 = 0B
37624  D1 ED B8 21 90 82 ED 4B = E1
37632  40 92 ED B8 3A 42 92 A7 = 2C
37640  28 B1 CD 0C 90 22 04 C0 = 28
37648  C3 10 8A 00 00 FE 63 C2 = 80
37656  3F 8C 1E 00 01 00 00 21 = 0B
37664  3F C0 7E FE FF 28 15 FE = B5
37672  21 30 05 1E 00 23 18 F2 = A1
37680  FE 81 30 F7 7B 3D 20 F5 = 7B
37688  03 1C 18 F1 ED 43 58 93 = 43
37696  11 5A 93 01 43 00 CD 3C = 4B
37704  20 ED 4B 58 93 CD 1B 1A = 45
37712  CD EB 85 28 FB C3 0A 8A = B7
37720  00 00 10 07 11 00 16 06 = 44
37728  09 20 20 20 20 20 20 20 = E9
37736  20 20 20 20 20 16 07 09 = C6
37744  20 57 6F 72 64 20 43 6F = 8E
37752  75 6E 74 20 16 08 09 20 = BE
37760  20 20 20 20 20 20 20 20 = 00
37768  20 20 20 16 09 09 20 20 = C8
37776  20 20 20 20 20 20 20 20 = 00
37784  20 20 16 08 0D 3D 3D 3D = 22
37792  3D 3D CA 6F 97 3D CA 43 = 94
37800  9B C3 80 95 13 01 14 01 = 9C
37808  16 02 02 20 48 45 4C 50 = 63
37816  2D 67 72 61 70 68 69 63 = 0B
37824  73 20 20 20 20 20 20 20 = 53
37832  FF 11 06 13 01 14 01 16 = 55
37840  03 02 54 14 00 63 6F 6D = AC
37848  70 72 65 73 73 65 64 20 = 16
37856  20 14 01 48 14 00 64 6F = 64
```

```
37864  75 62 6C 65 20 77 69 64 = 0C
37872  74 68 20 16 04 02 14 01 = 2D
37880  59 14 00 75 6E 64 65 72 = 8B
37888  6C 69 6E 65 20 20 20 14 = 1C
37896  01 55 14 00 69 74 61 6C = 14
37904  69 63 73 20 20 20 20 20 = DF
37912  20 16 05 02 14 01 49 14 = AF
37920  00 65 6D 70 68 61 73 69 = E7
37928  73 65 64 20 20 14 01 4F = E0
37936  14 00 64 6F 75 62 6C 65 = 8F
37944  20 73 74 72 69 6B 65 16 = C8
37952  06 02 14 01 47 14 00 73 = EB
37960  75 62 73 63 72 69 70 74 = 6C
37968  20 20 20 14 01 50 14 00 = D9
37976  73 75 70 65 72 73 63 72 = 77
37984  69 70 74 20 20 16 07 02 = AC
37992  14 01 4A 14 00 31 2F 38 = 0B
38000  74 68 20 73 70 61 63 65 = 08
38008  20 14 01 4B 14 00 37 2F = FA
38016  37 32 6E 64 20 73 70 61 = 9F
38024  63 65 20 16 08 02 14 01 = 1D
38032  4C 14 00 70 61 70 65 72 = 78
38040  20 73 65 6E 73 6F 72 20 = DA
38048  6F 6E 2F 6F 66 66 20 20 = 87
38056  20 20 20 20 20 16 09 02 = C1
38064  14 01 56 14 00 6C 69 6E = C2
38072  65 20 66 65 65 64 20 6F = A8
38080  6E 2F 6F 66 66 20 20 20 = 3E
38088  20 20 20 20 20 20 20 16 = F6
38096  0A 02 14 01 42 14 00 63 = DA
38104  6F 6C 75 6D 6E 20 77 69 = 2E
38112  64 74 68 20 42 3D 31 34 = 44
38120  32 2F 62 3D 38 30 20 20 = A8
38128  20 16 0B 02 14 01 4E 14 = BA
38136  00 64 6F 75 62 6C 65 20 = 9B
38144  77 69 64 74 68 20 31 20 = 91
38152  6C 69 6E 65 20 6F 6E 6C = 11
38160  79 20 20 16 0C 02 14 01 = F2
38168  4D 14 00 75 6E 69 64 69 = 7A
38176  72 65 63 74 69 6F 6E 61 = 55
38184  6C 20 70 72 69 6E 74 69 = 22
38192  6E 67 20 20 20 13 01 11 = 5A
38200  06 16 0D 02 14 01 20 55 = B5
38208  50 50 45 52 20 63 61 73 = 8E
38216  65 20 4F 4E 20 3A 20 6C = 08
38224  6F 77 65 72 20 6F 66 66 = 18
38232  20 FF 14 01 13 01 11 00 = 59
38240  10 02 7F 11 02 10 06 7F = 39
38248  11 06 10 04 7F 11 04 10 = CF
38256  05 7F 11 05 10 00 7F FF = 28
38264  10 00 11 07 20 14 00 FF = 5B
38272  3E 02 CD 01 16 21 AC 93 = 84
38280  CD 9F 95 CD A7 95 21 C9 = F4
38288  93 CD 9F 95 CD EB 85 28 = F9
38296  FB CD 74 9B C3 4B 8E 7E = F1
38304  FE FF C8 D7 23 18 F8 21 = F0
38312  36 5C 36 00 E5 21 5A 95 = C5
38320  CD 9F 95 E1 36 00 21 78 = B1
38328  95 CD 9F 95 C9 13 01 14 = 87
38336  01 16 02 02 20 48 45 4C = 14
38344  50 2D 74 65 78 74 20 20 = 82
38352  20 20 20 20 20 20 20 20 = 00
38360  20 FF 13 01 11 06 16 03 = 63
38368  02 20 20 20 20 20 20 20 = E2
38376  20 20 20 20 20 20 20 20 = 00
38384  20 20 20 20 20 20 20 20 = 00
38392  20 20 20 20 14 01 11 05 = AB
38400  16 04 02 41 52 52 4F 57 = A7
38408  53 2E 2E 2E 2E 14 00 63 = 82
38416  75 72 73 6F 72 20 63 6F = 2D
38424  6E 74 72 6F 6C 73 20 20 = E2
38432  14 01 16 05 02 44 45 4C = 07
38440  45 54 45 2E 2E 2E 2E 14 = AA
38448  00 64 65 6C 65 74 65 20 = 93
38456  63 68 61 72 61 63 74 65 = 3B
38464  72 20 14 00 16 06 02 20 = E4
38472  20 20 20 20 20 20 20 20 = 00
38480  20 14 00 62 65 68 69 6E = 3A
38488  64 20 63 75 72 73 6F 72 = 22
38496  20 20 20 20 14 01 16 07 = B2
38504  02 54 52 55 45 20 56 49 = 01
38512  44 45 4F 14 00 6E 65 78 = 37
38520  74 20 54 41 42 20 70 6F = 6A
38528  73 69 74 69 6F 6E 14 01 = AB
38536  16 08 02 49 4E 56 2E 20 = 5B
38544  56 49 44 45 4F 14 00 48 = D3
38552  45 4C 50 2D 74 65 78 74 = D3
38560  20 20 20 20 20 20 20 20 = 00
38568  14 01 16 09 02 47 52 41 = 10
38576  50 48 49 43 53 2E 2E 14 = E7
38584  00 70 72 69 6E 74 65 72 = 04
38592  20 63 6F 6E 74 72 6F 6C = 21
38600  73 20 14 01 16 0A 02 45 = 0F
38608  58 54 2E 20 4D 4F 44 45 = 1F
38616  2E 14 00 74 65 78 74 20 = 27
38624  66 6F 72 6D 61 74 20 20 = C9
38632  20 20 20 20 14 01 16 0B = B6
38640  02 45 44 49 54 2E 2E 2E = B2
38648  2E 2E 2E 14 00 72 65 74 = E9
38656  75 72 6E 20 74 6F 20 4D = C5
38664  45 4E 55 20 20 20 14 00 = 5C
38672  11 06 16 0C 02 20 46 6F = 10
38680  72 20 47 52 41 50 48 49 = 4D
38688  43 53 2F 45 2E 4D 4F 44 = 18
38696  45 20 48 45 4C 50 20 20 = CE
38704  16 0D 02 20 20 73 65 6C = A9
38712  65 63 74 20 4B 45 59 20 = 65
38720  26 20 49 4E 56 2E 20 56 = D7
38728  49 44 45 4F 20 20 16 0E = 85
38736  02 14 01 20 20 20 50 72 = 39
38744  65 73 73 20 45 4E 54 45 = 97
38752  52 20 66 6F 72 20 54 45 = 72
38760  58 54 20 20 20 20 FF 3E = 69
38768  02 CD 01 16 21 BD 95 CD = 26
38776  9F 95 CD A7 95 21 DA 95 = CD
38784  CD 9F 95 CD EB 85 20 FB = 61
38792  C3 0A 8A 13 01 14 01 16 = 96
38800  02 02 20 48 45 4C 50 2D = 7A
38808  45 78 74 20 6D 6F 64 65 = F6
38816  20 20 20 20 20 20 20 FF = DF
38824  13 00 11 04 10 00 16 03 = 51
38832  02 20 41 52 52 4F 57 53 = 00
38840  20 20 20 20 20 47 6F 74 = CA
38848  6F 20 74 65 78 74 3A 2D = BB
38856  20 20 20 20 16 04 02 13 = AF
38864  01 20 55 50 20 20 20 20 = 46
38872  20 20 20 3A 20 53 74 61 = E2
```

```
38880  72 74 20 20 20 20 20 20 = A6
38888  20 20 20 20 16 05 02 20 = BD
38896  44 4F 57 4E 20 20 20 20 = B8
38904  20 3A 20 45 6E 64 20 20 = D1
38912  20 20 20 20 20 20 20 20 = 00
38920  20 20 16 06 02 20 52 49 = 19
38928  47 48 54 20 20 20 20 3A = 9D
38936  20 52 69 67 68 74 20 20 = 5E
38944  6D 61 72 67 69 6E 20 20 = BE
38952  16 07 02 20 4C 45 46 54 = 6A
38960  20 20 20 20 20 3A 20 4C = 46
38968  65 66 74 20 20 20 6D 61 = 6D
38976  72 67 69 6E 20 20 13 01 = 04
38984  11 05 16 08 02 20 20 20 = 96
38992  51 20 3C 3D 20 20 53 45 = C2
39000  54 20 6C 65 66 74 20 20 = 5F
39008  6D 61 72 67 69 6E 20 20 = BE
39016  16 09 02 20 20 20 45 20 = E6
39024  3E 3D 20 20 53 45 54 20 = C7
39032  72 69 67 68 74 20 6D 61 = 0C
39040  72 67 69 6E 20 20 16 0A = 10
39048  02 20 20 20 52 20 3C 20 = 30
39056  52 45 53 45 54 20 6C 65 = 74
39064  66 74 20 20 6D 61 72 67 = C1
39072  69 6E 20 20 16 0B 02 20 = 5A
39080  20 20 54 20 3E 20 52 45 = A9
39088  53 45 54 20 72 69 67 68 = B6
39096  74 20 6D 61 72 67 69 6E = 12
39104  20 20 13 01 11 06 16 0C = 8D
39112  02 49 3A 49 6E 73 65 72 = 86
39120  74 20 6D 6F 64 65 20 20 = 79
39128  11 04 43 3A 57 6F 72 64 = 2E
39136  20 63 6F 75 6E 74 16 0D = 6C
39144  02 11 06 4F 3A 4F 76 65 = CC
39152  72 70 72 6E 74 20 6D 6F = 32
39160  64 65 11 04 57 3A 57 6F = 35
39168  72 64 20 66 69 6E 64 20 = B7
39176  16 0E 02 11 06 4B 3A 57 = 19
39184  6F 72 64 77 72 61 70 20 = 1F
39192  4F 4E 20 20 11 04 20 20 = 32
39200  20 20 20 20 20 20 20 20 = 00
39208  20 20 16 0F 02 11 06 4A = C8
39216  3A 57 6F 72 64 77 72 61 = 20
39224  70 20 4F 46 46 20 11 04 = A0
39232  20 20 20 20 20 20 20 20 = 00
39240  20 20 20 20 16 10 02 11 = B9
39248  02 14 01 20 20 59 2F 6E = 4D
39256  65 78 74 20 70 61 67 65 = 0E
39264  20 3A 20 20 4E 2F 72 65 = EE
39272  74 75 72 6E 20 20 FF 13 = 1B
39280  01 14 01 16 02 02 20 48 = 98
39288  45 4C 50 2D 45 78 74 20 = 5F
39296  6D 6F 64 65 20 63 6F 6E = 05
39304  74 2E 20 FF 13 00 11 04 = E9
39312  10 00 16 03 02 20 20 20 = 8B
39320  20 20 20 20 54 41 42 20 = 77
39328  20 43 6F 6D 6D 61 6E 64 = DF
39336  73 20 20 20 20 20 20 20 = 53
39344  16 04 02 13 01 20 4B 65 = 00
39352  79 20 31 20 20 20 3A 20 = 84
39360  53 45 54 20 74 61 62 20 = 63
39368  20 20 20 20 20 20 20 20 = 00
39376  16 05 02 20 4B 65 79 20 = 86
39384  32 20 20 20 3A 20 52 45 = 83
39392  53 45 54 20 74 61 62 20 = 63
39400  20 20 20 20 20 20 16 06 = DC
39408  02 20 4B 65 79 20 33 20 = BE
39416  20 20 3A 20 52 45 53 45 = C9
39424  54 20 41 4C 4C 20 74 61 = 42
39432  62 73 20 20 16 07 02 13 = 47
39440  00 11 05 20 20 20 20 20 = B6
39448  20 42 4C 4F 43 4B 20 20 = CB
39456  43 6F 6D 6D 61 6E 64 73 = 32
39464  20 20 20 20 20 20 13 01 = D4
39472  16 08 02 20 4B 65 79 20 = 89
39480  38 20 20 20 3A 20 53 45 = 8A
39488  54 20 62 6C 6F 63 6B 20 = 9F
39496  73 74 61 72 74 20 16 09 = 6D
39504  02 20 4B 65 79 20 39 20 = C4
39512  20 20 3A 20 53 45 54 20 = A6
39520  62 6C 6F 63 6B 20 65 6E = FE
39528  64 20 20 20 16 0A 02 20 = 06
39536  4B 65 79 20 30 20 20 20 = D9
39544  3A 20 52 45 53 45 54 20 = FD
39552  62 6C 6F 63 6B 73 20 20 = BE
39560  20 20 16 0B 02 20 4B 65 = 33
39568  79 20 35 20 20 20 3A 20 = 88
39576  43 4F 50 59 20 62 6C 6F = 98
39584  63 6B 20 20 20 20 20 20 = 8E
39592  16 0C 02 20 4B 65 79 20 = 8D
39600  36 20 20 20 3A 20 4D 4F = 8C
39608  56 45 20 62 6C 6F 63 6B = C6
39616  20 20 20 20 20 20 16 0D = E3
39624  02 20 4B 65 79 20 37 20 = C2
39632  20 20 3A 20 44 45 4C 45 = B4
39640  54 45 20 62 6C 6F 63 6B = C4
39648  20 20 20 20 16 0E 02 11 = B7
39656  06 20 20 20 20 20 20 20 = E6
39664  20 20 20 20 20 20 20 20 = 00
39672  20 20 20 20 20 20 20 20 = 00
39680  20 20 20 20 16 0F 02 20 = C7
39688  20 20 20 20 20 20 20 20 = 00
39696  20 20 20 20 20 20 20 20 = 00
39704  20 20 20 20 20 20 20 20 = 00
39712  20 20 16 10 02 14 01 59 = D6
39720  2F 50 72 65 76 69 6F 75 = 19
39728  73 20 70 61 67 65 20 3A = 8A
39736  20 20 4E 2F 72 65 74 75 = 7D
39744  72 6E FF 3E 02 CD 01 16 = 03
39752  21 8B 97 CD 9F 95 CD A7 = B8
39760  95 21 A8 97 CD 9F 95 CD = C3
39768  EB 85 28 FB FE 4E 28 0E = 15
39776  FE 6E 28 0A FE 59 28 2E = 4B
39784  FE 79 28 2A 18 E9 CD 74 = 0B
39792  9B C3 0F 8D 21 20 58 01 = 94
39800  DF 02 11 21 58 3A 17 C0 = 7C
39808  77 ED B0 21 60 5A 01 1F = 0F
39816  00 11 61 5A 3A 18 C0 77 = 55
39824  ED B0 CD 83 87 C9 3E 02 = 7D
39832  CD 01 16 21 6F 99 CD 9F = 79
39840  95 CD A7 95 21 8C 99 CD = B1
39848  9F 95 CD EB 85 20 FB FE = 92
39856  4E 28 BB FE 6E 28 B7 FE = 7A
39864  59 28 88 FE 79 28 84 18 = 44
39872  E9 3E 02 CD 01 16 21 D9 = 07
39880  9B CD 9F 95 CD A7 95 21 = C6
39888  F5 9B CD 9F 95 CD A7 95 = 9A
```

```
39896 C9 16 05 03 13 01 14 01 = 10
39904 20 57 6F 72 64 20 66 69 = AB
39912 6E 64 20 20 20 20 20 20 = 92
39920 20 20 20 20 FF 13 01 11 = A4
39928 06 16 06 03 14 00 20 45 = 9E
39936 6E 74 65 72 20 63 68 61 = 05
39944 72 61 63 74 65 72 73 20 = 14
39952 69 6E 20 6C 6F 77 65 72 = 20
39960 16 07 03 20 63 61 73 65 = DC
39968 20 74 6F 20 66 69 6E 64 = C4
39976 20 77 6F 72 64 73 20 69 = D8
39984 6E 20 20 20 20 16 08 03 = 0F
39992 20 65 69 74 68 65 72 20 = C1
40000 75 70 70 65 72 20 6F 72 = 2D
40008 20 6C 6F 77 65 72 20 20 = 89
40016 20 20 16 09 03 20 63 61 = 46
40024 73 65 2E 20 4F 72 20 75 = 7C
40032 73 65 20 55 50 50 45 52 = 84
40040 20 63 61 73 65 20 20 16 = 12
40048 0A 03 20 74 6F 20 66 69 = FF
40056 6E 64 20 55 50 50 45 52 = 7E
40064 20 63 61 73 65 20 77 6F = C2
40072 72 64 73 2E 16 0B 03 20 = BB
40080 20 20 20 20 20 20 20 20 = 00
40088 20 20 20 20 20 20 20 20 = 00
40096 20 20 20 20 20 20 20 20 = 00
40104 20 16 0C 03 20 50 72 65 = 8C
40112 73 73 20 45 4E 54 45 52 = 84
40120 20 74 6F 20 66 69 6E 64 = C4
40128 20 6C 61 73 74 20 16 0D = 17
40136 03 20 77 6F 72 64 20 73 = 72
40144 65 61 72 63 68 65 64 20 = EC
40152 66 6F 72 2E 20 20 20 20 = F5
40160 20 20 20 16 15 00 13 01 = 9F
40168 14 01 20 45 6E 74 65 72 = 33
40176 20 77 6F 72 64 20 6F 72 = DD
40184 20 70 68 72 61 73 65 20 = C3
40192 20 20 20 20 FF 3E 02 CD = 8C
40200 01 16 21 1A 9D CD 9F 95 = F0
40208 CD A7 95 21 32 9D CD 9F = 65
40216 95 C9 14 01 13 01 16 05 = A2
40224 05 20 43 75 73 74 6F 6D = A0
40232 69 7A 65 20 20 20 20 20 = E8
40240 20 FF 14 00 16 06 05 20 = 74
40248 54 65 78 74 20 76 61 6C = 08
40256 75 65 20 20 20 20 20 20 = 9A
40264 20 20 20 20 20 16 07 05 = C2
40272 20 57 69 6E 64 6F 77 20 = B8
40280 76 61 6C 75 65 20 20 20 = 7D
40288 20 20 20 20 20 20 16 08 = DE
40296 05 20 42 6F 72 64 65 72 = 83
40304 20 76 61 6C 75 65 20 20 = 7D
40312 20 20 20 20 20 20 20 16 = F6
40320 09 05 20 60 20 73 69 67 = F1
40328 6E 20 63 6F 64 65 20 20 = 69
40336 20 20 20 20 20 20 20 20 = 00
40344 16 0A 05 20 45 78 69 74 = DF
40352 20 20 20 20 20 20 20 20 = 00
40360 20 20 20 20 20 20 20 20 = 00
40368 20 FF 3E 02 CD 01 16 21 = 64
40376 C7 9D CD 9F 95 CD A7 95 = 6E
40384 21 E5 9D CD 9F 95 C9 13 = 80
40392 01 14 01 16 03 02 20 46 = 97
40400 6F 72 6D 61 74 20 20 20 = 83
40408 20 20 20 20 20 20 20 20 = 00
40416 20 20 20 20 FF 14 00 13 = A6
40424 01 16 04 02 20 4C 69 6E = 60
40432 65 73 20 70 65 72 20 70 = CF
40440 61 67 65 20 20 20 20 20 = CD
40448 20 20 20 20 20 20 20 20 = 00
40456 16 05 02 20 53 70 61 63 = C4
40464 65 73 20 62 65 74 77 65 = 0F
40472 65 6E 20 6C 69 6E 65 73 = 0E
40480 20 20 20 20 20 20 20 16 = F6
40488 06 02 20 4D 61 72 67 69 = 10
40496 6E 20 20 20 20 20 20 20 = 4E
40504 20 20 20 20 20 20 20 20 = 00
40512 20 20 20 20 20 20 16 07 = DD
40520 02 20 53 74 61 72 74 20 = 50
40528 61 74 20 70 61 67 65 20 = B2
40536 4E 6F 2E 20 20 20 20 20 = 8B
40544 20 20 20 20 20 16 08 02 = C0
40552 20 50 61 67 65 20 4E 6F = 7A
40560 2E 20 61 74 20 54 4F 50 = 36
40568 20 20 20 20 20 20 20 20 = 00
40576 20 20 20 20 16 09 02 20 = C1
40584 50 61 67 65 20 4E 6F 2E = 88
40592 20 61 74 20 42 4F 54 2E = 28
40600 20 20 20 20 20 20 20 20 = 00
40608 20 20 20 16 0A 02 20 57 = F9
40616 61 69 74 20 62 65 74 77 = 10
40624 65 65 6E 20 70 61 67 65 = F5
40632 73 20 20 20 20 20 20 20 = 53
40640 20 20 16 0B 02 20 41 75 = 39
40648 74 6F 20 6C 69 6E 65 20 = CB
40656 66 65 65 64 20 20 20 20 = 14
40664 20 20 20 20 20 20 20 20 = 00
40672 20 16 0C 02 20 45 78 69 = 8A
40680 74 20 20 20 20 20 20 20 = 54
40688 20 20 20 20 20 20 20 20 = 00
40696 20 20 20 20 20 20 20 20 = 00
40704 FF 3E 02 CD 01 16 21 10 = 54
40712 9F CD 9F 95 CD A7 95 C9 = 72
40720 16 15 00 13 01 14 01 11 = 65
40728 07 10 00 20 53 70 65 63 = C2
40736 77 6F 72 64 20 31 32 38 = 77
40744 20 20 20 20 20 20 20 20 = 00
40752 20 20 20 20 20 FF 3E 02 = DF
40760 CD 01 16 CD A7 95 C9 01 = B7
40768 00 00 3A 48 00 FE 6E C0 = B1
40776 3A 38 5C CB 67 C0 03 C9 = 97
40784 3E 02 CD 01 16 21 65 9F = 49
40792 CD 9F 95 CD A7 95 21 77 = A2
40800 9F CD 9F 95 C9 14 01 13 = 91
40808 01 16 05 07 20 44 6F 63 = 59
40816 75 6D 65 6E 74 20 FF 13 = 5B
40824 01 14 00 16 06 07 20 20 = 78
40832 20 43 72 65 61 74 65 20 = 94
40840 20 20 20 20 20 20 16 07 = DD
40848 07 20 20 20 4C 6F 61 64 = E7
40856 20 20 20 20 20 20 20 20 = 00
40864 20 16 08 07 20 20 20 53 = F8
40872 61 76 65 20 20 20 20 20 = DC
40880 20 20 20 20 16 09 07 20 = C6
40888 20 20 50 72 69 6E 74 20 = 6D
40896 20 20 20 20 20 20 20 16 = F6
40904 0A 07 20 20 20 45 64 69 = 83
40912 74 20 20 20 20 20 20 20 = 54
40920 20 20 16 0B 07 20 20 20 = C8
40928 43 41 54 21 20 20 20 20 = 79
40936 20 20 20 20 20 16 0C 07 = C9
40944 20 20 20 43 75 73 74 6F = 6E
40952 6D 69 7A 65 20 20 20 20 = 35
40960 FF 3E 02 CD 01 16 21 16 = 5A
40968 A0 CD 9F 95 CD A7 95 21 = CB
40976 28 A0 CD 9F 95 C9 13 01 = A6
40984 14 01 16 05 07 20 43 72 = 0C
40992 65 61 74 65 20 20 20 FF = FE
41000 13 01 14 00 16 06 07 20 = 6B
41008 20 20 43 6F 6E 74 69 6E = AB
41016 75 65 20 20 20 20 20 16 = 90
41024 07 07 20 20 20 45 78 69 = 94
41032 74 20 20 20 20 20 20 20 = 54
41040 20 20 FF 3E 02 CD 01 16 = 63
41048 21 68 A0 CD 9F 95 CD A7 = 9E
41056 95 21 7A A0 CD 9F 95 C9 = 9A
41064 13 01 14 01 16 05 07 20 = 6B
41072 4C 6F 61 64 20 20 20 20 = 00
41080 20 FF 13 01 14 00 16 06 = 63
41088 07 20 20 20 54 61 70 65 = F1
41096 20 20 20 20 20 20 20 20 = 00
41104 20 16 07 07 20 20 20 52 = F6
41112 41 4D 21 20 20 20 20 20 = 4F
41120 20 20 20 20 16 08 07 20 = C5
41128 20 20 45 78 69 74 20 20 = 1A
41136 20 20 20 20 20 20 20 FF = DF
41144 3E 02 CD 01 16 21 CD A0 = B2
41152 CD 9F 95 CD A7 95 21 DF = 0A
41160 A0 CD 9F 95 C9 13 01 14 = 92
41168 01 16 05 07 20 53 61 76 = 6D
41176 65 20 20 20 20 20 FF 13 = 17
41184 01 14 00 16 06 07 20 20 = 78
41192 20 54 61 70 65 20 20 20 = 0A
41200 20 20 20 20 20 20 16 07 = DD
41208 07 20 20 20 52 41 4D 21 = 68
41216 20 20 20 20 20 20 20 20 = 00
41224 20 16 08 07 20 20 20 52 = F7
41232 65 74 69 74 6C 65 20 20 = C7
41240 20 20 20 20 16 09 07 20 = C6
41248 20 20 50 72 6F 67 72 61 = AB
41256 6D 20 20 20 20 20 20 16 = 43
41264 0A 07 20 20 20 45 78 69 = 97
41272 74 20 20 20 20 20 20 20 = 54
41280 20 20 FF 3E 02 CD 01 16 = 63
41288 21 58 A1 CD 9F 95 CD A7 = 8F
41296 95 21 6A A1 CD 9F 95 C9 = 8B
41304 13 01 14 01 16 05 07 20 = 6B
41312 50 72 69 6E 74 20 20 20 = 6D
41320 20 FF 13 01 14 00 16 06 = 63
41328 07 20 20 20 41 6C 6C 20 = A0
41336 74 65 78 74 20 20 20 20 = 45
41344 20 16 07 07 20 20 20 42 = E6
41352 6C 6F 63 6B 20 20 20 20 = 29
41360 20 20 20 20 16 08 07 20 = C5
41368 20 20 46 6F 72 6D 61 74 = A9
41376 20 20 20 20 20 20 20 16 = F6
41384 09 07 20 20 20 43 6F 64 = 86
41392 65 73 20 20 20 20 20 20 = 98
41400 20 20 16 0A 07 20 20 20 = C7
41408 49 6E 69 74 69 61 6C 69 = 33
41416 73 65 20 20 20 16 0B 07 = 60
41424 20 20 20 45 78 69 74 20 = 1A
41432 20 20 20 20 20 20 20 20 = 00
41440 FF 3E 02 CD 01 16 21 F6 = 3A
41448 A1 CD 9F 95 CD A7 95 21 = CC
41456 08 A2 CD 9F 95 C9 14 01 = 89
41464 13 01 16 02 07 20 43 6F = 05
41472 64 65 73 20 20 20 20 FF = BB
41480 14 00 13 01 16 03 07 20 = 68
41488 20 20 20 20 20 42 2F 2F = 40
41496 62 20 20 20 20 20 20 16 = 38
41504 04 07 20 20 20 20 20 20 = CB
41512 47 2F 2F 67 20 20 20 20 = 8C
41520 20 20 16 05 07 20 20 20 = C2
41528 20 20 20 48 2F 2F 68 20 = 8E
41536 20 20 20 20 20 16 06 07 = C3
41544 20 20 20 20 20 20 49 2F = 38
41552 2F 69 20 20 20 20 20 20 = 58
41560 16 07 07 20 20 20 20 20 = C4
41568 20 4A 2F 2F 6A 20 20 20 = 92
41576 20 20 20 16 08 07 20 20 = C5
41584 20 20 20 20 4B 2F 2F 6B = 94
41592 20 20 20 20 20 20 16 09 = DF
41600 07 20 20 20 20 20 20 4C = 13
41608 2F 2F 6C 20 20 20 20 20 = 6A
41616 20 16 0A 07 20 20 20 20 = C7
41624 20 20 4D 2F 2F 6D 20 20 = 98
41632 20 20 20 20 16 0B 07 20 = C8
41640 20 20 20 20 20 4E 2F 2F = 4C
41648 6E 20 20 20 20 20 20 16 = 44
41656 0C 07 20 20 20 20 20 20 = D3
41664 4F 2F 2F 6F 20 20 20 20 = 9C
41672 20 20 16 0D 07 20 20 20 = CA
41680 20 20 20 50 2F 2F 70 20 = 9E
41688 20 20 20 20 20 16 0E 07 = CB
41696 20 20 20 20 20 20 54 2F = 43
41704 2F 74 20 20 20 20 20 20 = A3
41712 16 0F 07 20 20 20 20 20 = CC
41720 20 55 2F 2F 75 20 20 20 = A8
41728 20 20 20 16 10 07 20 20 = CD
41736 20 20 20 20 56 2F 2F 76 = AA
41744 20 20 20 20 20 20 16 11 = E7
41752 07 20 20 20 20 20 20 59 = 20
41760 2F 2F 79 20 20 20 20 20 = 77
41768 20 16 12 07 20 20 20 20 = CF
41776 20 20 45 78 69 74 20 20 = 1A
41784 20 20 20 20 FF 3E 02 CD = 8C
41792 01 16 21 52 A3 CD 9F 95 = 2E
41800 CD A7 95 21 64 A3 CD 9F = 9D
41808 95 C9 14 01 13 01 16 05 = A2
41816 07 20 43 61 73 65 20 20 = E3
41824 20 20 20 FF 14 00 13 01 = 87
41832 16 06 07 20 20 20 55 70 = 48
41840 70 65 72 20 20 20 20 20 = E7
41848 20 20 20 16 07 07 20 20 = C4
41856 20 4C 6F 77 65 72 20 20 = 69
41864 20 20 20 20 20 20 16 08 = DE
41872 07 20 20 20 45 78 69 74 = 01
41880 20 20 20 20 20 20 20 20 = 00
41888 20 FF 3E 02 CD 01 16 21 = 64
41896 AE A3 CD 9F 95 C9 14 01 = 30
41904 13 01 16 15 00 20 54 6F = 22
41912 6F 20 6D 61 6E 79 20 63 = C7
41920 68 61 72 61 63 74 65 72 = 4A
41928 73 20 20 20 20 20 20 FF = 32
41936 3E 02 CD 01 16 21 DC A3 = C4
41944 CD 9F 95 C9 14 01 13 01 = F3
41952 16 15 00 20 49 6E 76 61 = D9
41960 6C 69 64 20 54 69 74 6C = F6
41968 65 20 20 20 20 20 20 20 = 45
41976 20 20 20 20 20 FF 3E 02 = DF
41984 CD 01 16 21 0A A4 CD 9F = 1F
41992 95 C9 14 01 13 01 16 15 = B2
42000 00 20 44 6F 63 75 6D 65 = 7D
42008 6E 74 20 4D 55 53 54 20 = 6B
42016 68 61 76 65 20 61 20 6E = B3
42024 61 6D 65 FF 3E 02 CD 01 = 40
42032 16 21 38 A4 CD 9F 95 C9 = DD
42040 14 01 13 01 16 15 00 20 = 74
42048 53 70 65 63 77 6F 72 64 = 47
42056 20 34 38 2F 28 20 20 20 = 46
42064 20 20 20 20 20 20 20 20 = 00
42072 20 FF 3E 02 CD 01 16 21 = 64
42080 66 A4 CD 9F 95 C9 14 01 = E9
42088 13 01 16 15 00 20 43 61 = 03
42096 74 61 6C 6F 67 75 65 20 = 11
42104 6F 66 20 52 41 4D 21 20 = 16
42112 46 69 6C 65 73 20 20 FF = 32
42120 3E 02 CD 01 16 21 94 A4 = 7D
42128 CD 9F 95 C9 14 01 13 01 = F3
42136 16 15 00 20 50 72 65 73 = E5
42144 73 20 61 6E 79 20 6B 65 = CB
42152 79 20 20 20 20 20 20 20 = 59
42160 20 20 20 20 20 FF 3E 02 = DF
42168 CD 01 16 21 C2 A4 CD 9F = D7
42176 95 C9 14 01 13 01 16 15 = B2
42184 00 20 50 72 69 6E 74 65 = 92
42192 72 20 6E 6F 77 20 69 6E = DD
42200 69 74 69 61 6C 69 73 65 = 54
42208 64 20 20 FF 3E 02 CD 01 = B1
42216 16 21 F0 A4 CD 9F 95 C9 = 95
42224 14 01 13 01 16 15 00 20 = 74
42232 54 69 74 6C 65 20 66 6F = F7
42240 72 20 4C 6F 61 64 69 6E = E9
42248 67 20 3F 20 20 20 20 20 = 66
42256 FF 3E 02 CD 01 16 21 1D = 61
42264 A5 CD 9F 95 C9 14 01 13 = 97
42272 01 16 15 00 20 56 65 72 = 79
42280 69 66 79 20 3F 20 3C 59 = 5C
42288 3E 3C 4E 3E 20 20 20 20 = 86
42296 20 20 20 20 20 FF 3E 02 = DF
42304 CD 01 16 21 4A A5 CD 9F = 60
42312 95 C9 14 01 13 01 16 15 = B2
42320 00 20 56 65 72 69 66 79 = 95
42328 20 4F 2E 4B 2E 20 20 20 = 76
42336 20 20 20 20 20 20 20 20 = 00
42344 20 20 FF 3E 02 CD 01 16 = 63
42352 21 77 A5 CD 9F 95 C9 14 = 1B
42360 01 13 01 16 15 00 20 50 = B0
42368 72 69 6E 74 65 72 20 4F = 03
42376 4E 20 4C 49 4E 45 3F 20 = F5
42384 3C 45 4E 54 45 52 3E FF = F7
42392 3E 02 CD 01 16 21 A4 A5 = 8E
42400 CD 9F 95 C9 14 01 13 01 = F3
42408 16 15 00 20 50 72 65 73 = E5
42416 73 20 45 4E 54 45 52 20 = 31
42424 74 6F 20 63 6F 6E 74 69 = 20
42432 6E 75 65 20 FF 3E 02 CD = 74
42440 01 16 21 D1 A5 CD 9F 95 = AF
42448 C9 14 01 13 01 16 15 00 = 1D
42456 20 4E 6F 74 20 61 76 61 = A9
42464 69 6C 61 62 6C 65 20 69 = F2
42472 6E 20 34 38 4B 20 6D 6F = 41
42480 64 65 FF 3E 02 CD 01 16 = EC
42488 21 FF A5 CD 9F 95 C9 14 = A3
42496 01 13 01 16 15 00 20 44 = A4
42504 6F 63 75 6D 65 6E 74 20 = 1B
42512 74 69 74 6C 65 20 3F 20 = A1
42520 20 20 20 20 20 20 20 20 = 00
42528 FF 3E 02 CD 01 16 21 3B = 7F
42536 A6 CD 9F 95 FD CB 01 6E = DE
42544 FD CB 01 6E 28 FA FD CB = 21
42552 01 AE C9 11 07 10 01 16 = B7
42560 05 07 53 50 45 43 57 4F = DD
42568 52 44 20 20 34 38 2F 28 = 9C
42576 2F 31 32 38 16 07 07 20 = 0E
42584 20 57 6F 72 64 20 50 72 = 9E
42592 6F 63 65 73 73 6F 72 20 = 1E
42600 20 16 09 07 20 20 20 56 = FC
42608 65 72 73 69 6F 6E 20 20 = D0
42616 31 2E 31 20 20 20 16 0B = 11
42624 05 7F 20 53 74 75 61 72 = B3
42632 74 20 4E 69 63 68 6F 6C = F1
42640 6C 73 20 31 39 38 36 FF = D6
42648 3E 02 CD 01 16 21 A4 A6 = 8F
42656 CD 9F 95 C9 14 01 13 01 = F3
42664 16 15 00 20 4E 6F 20 74 = 9C
42672 65 78 74 20 66 69 6C 65 = 11
42680 20 74 6F 20 45 44 49 54 = 49
42688 20 20 20 20 20 FF 3A 10 = E9
42696 C0 EE 01 C6 24 32 76 91 = D2
42704 3A 16 C0 C6 24 32 71 91 = 2E
42712 3A 6A 5C E6 08 CB 3F CB = C3
42720 3F CB 3F C6 24 32 6D 91 = 63
42728 21 7D 91 06 05 36 30 23 = C3
42736 10 FB 21 88 91 06 05 36 = 86
42744 30 23 10 FB ED 4B 12 C0 = 68
42752 78 B1 28 17 03 21 81 91 = 9E
42760 78 B1 28 0F 7E 3C FE 3A = 52
42768 20 05 36 30 2B 18 F5 77 = 3A
42776 0B 18 EA ED 4B 14 C0 78 = 91
42784 B1 C8 03 21 8C 91 78 B1 = E3
42792 C8 7E 3C FE 3A 20 05 36 = 15
42800 30 2B 18 F5 77 0B 18 EB = ED
42808 FE 31 C2 1C 8A 21 00 40 = F8
42816 11 01 40 01 FF 17 36 00 = 9F
42824 ED B0 ED 5B 00 C0 1B 21 = E1
42832 20 41 E5 06 20 CD AC A7 = 8C
42840 28 02 CB FE CD AC A7 28 = 38
42848 02 CB EE CD AC A7 28 02 = 05
42856 CB DE CD AC A7 28 02 CB = BE
42864 CE 23 10 E1 E1 7D C6 20 = 26
42872 6F 20 D7 7C C6 08 67 FE = 15
42880 59 20 CF 21 00 41 11 00 = 88
42888 42 01 FF 02 ED B0 21 00 = 02
42896 49 11 00 4A 01 FF 02 ED = 93
42904 B0 21 00 51 11 00 52 01 = 86
42912 FF 02 ED B0 CD EB 85 28 = 03
42920 FB C3 0A 8A 13 1A FE 80 = FD
42928 38 FA FE 0D 28 05 FE 20 = 80
42936 38 F2 C9 E1 18 B6 FF = A1
```

STARGLIDER

A classic game from Rainbird

Starglider
Rainbird
£14.95

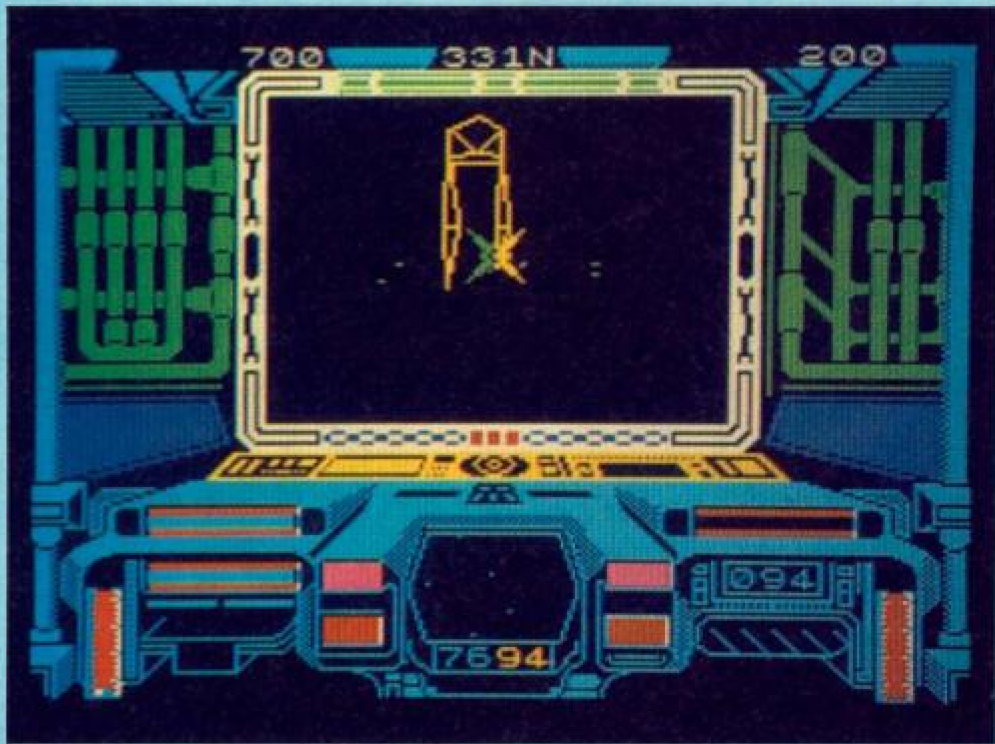
"HEALTH WARNING: DON'T MESS WITH NOVENIA" was the message written on the side of the deadly Sentinels that destroyed anything that threatened the peace loving people on the planet below.

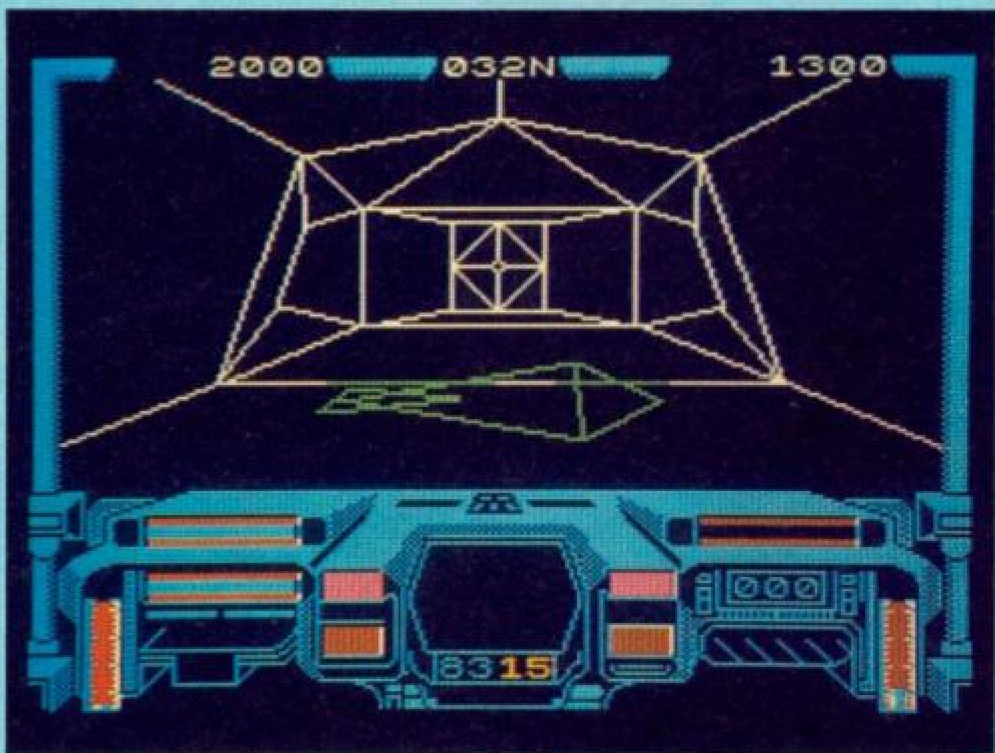
The invading Egrons learnt this to their cost as they lost countless ships to the fleet mangling defences. If the Sentinels didn't like the look of something that something rapidly became scrap.

Unfortunately the peace loving Novenians were also conservationists and modified the Sentinels to let the Starglider birds migrate without being fried in the process. So when the Egrons sent a Starglider shaped fleet to attack Novenia the Sentinels let them in.

Now you play two teenage heroes who have found a prototype AGAV (airborne ground attack vehicle) and set off to defeat the Egrons who now use Novenia's defences against you!

The full plot that leads to this improbable situation is described in a 64 page novella that accompanies the tape containing both 48K and 128K versions as well as a playguide that introduces the AGAV's controls, a keyguide to show you which buttons to use and poster of your AGAV. Not bad, even for £14.95! That's twice the average cost of a Spectrum game but Starglider is no ordinary game.





Stomp and walk

Graphically it will be compared to Elite as it features the same 3D vector line graphics that represent the tanks, walkers, stompers, missile launchers and Stargliders of the defence forces that now home in on you.

Your mission is simply to get them before they get you.

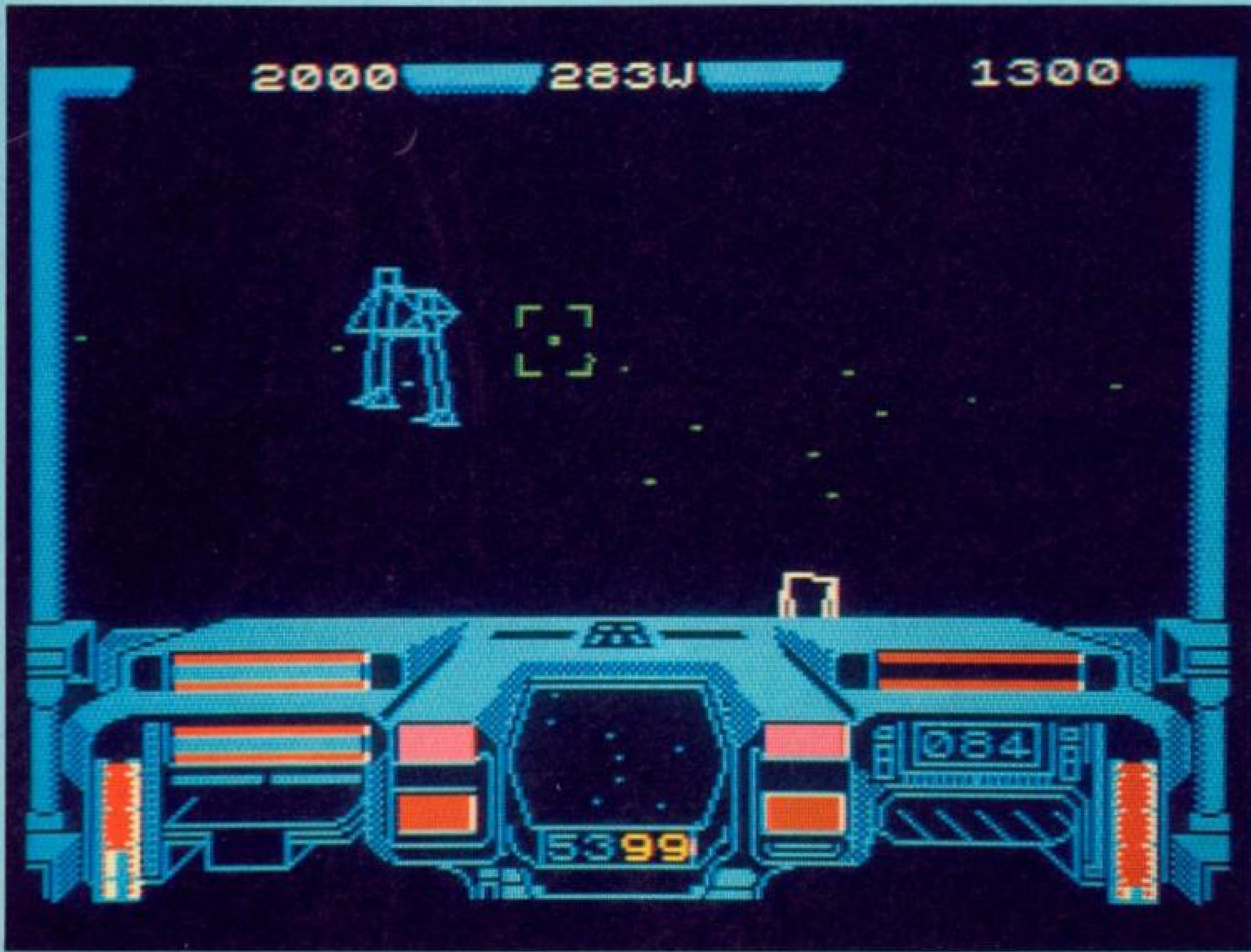
There's no great strategic aim to the game except to amass as many points as possible and so wreak revenge on the dreaded Ergons.

At first you'll tackle the light tanks that trundle across the planet's surface with your "Saphire II" quadpulse laser but at only 20 points each you'll soon be looking for bigger prey but since even these tanks fire ship wrecking missiles they must be taken seriously.

Using your scanner set in the middle of your control panel you can track the movement and postion of the enemy defences. After a few practise missions you'll be able to take on the missile launchers that greet you with a hail of homing missiles, the Star Wars inspired walkers and stomping stompers that can only be destroyed by a well guided missile and the Stargliders that fly with a graceful flap of their mechanical wings and can deliver a killing laser bolt.

In the silo

All this takes its toll on your fuel,







laser power, shields and meagre missile supply but luckily you can use some of the planet's installations. Docking with an Alliance space station repair silo will bring a welcome breather as well as repairs and even a special mission. To refuel you must fly at low level between the twin towers of the plasma energy station and run along its lines and pull out when you reach the third single tower and you will have absorbed enough energy to continue your game.

Starglider was written for Rainbird by 3D experts, Realtime Software (3D Starstrike) and it's their best game yet. It was originally released on the Atari ST which is almost identical to the 128K Spectrum version with its impressive music and digitised speech! The 48K version has the same action packed gameplay but without these added features. A must for all Spectrum owners.



MARBLE MADNESS CONSTRUCTION SET

Melbourne House release a 'customise a coin-op classic' package.

Marble Madness Construction Set
Melbourne House
£8.95



Marble Madness, the coin-op game was an original that like all good ideas has spawned numerous clone games in the same vein. Gyroscope (Melbourne House) and Spindizzy (Electric Dreams) have already appeared and the official Marble Madness game was released on the C64. But what of the Spectrum?

Implementation problems mean that a pure conversion is out of the question. So if you are going to have to change the game a bit, why not change it a lot? The Melbourne House conversion, whilst retaining all of the spirit and playability of the original game, also has a lot of extras (and very few omissions). So much extra has been crammed in that you soon forget about the graphical differences with the original.

The game is based on rolling a marble down an isometric 3D-ish landscape that has holes, slippery areas, barriers, lifts, drains, conveyor belts, vacuum cleaners, slime monsters, enemy marbles, bonus areas, acid, and of course the goal — which is the object of all your efforts. In the Melbourne House variant of the game some of these hindrances are missing — but it doesn't seem to help you any. Also the level system is worked out rather differently. Each level is composed of a static screen. Instead of smooth scrolling a long level up, the screen stays stationary and shifts up when you get to the bottom. After eleven of these screens you go back to the beginning and start again, but taking your time bonus with you.

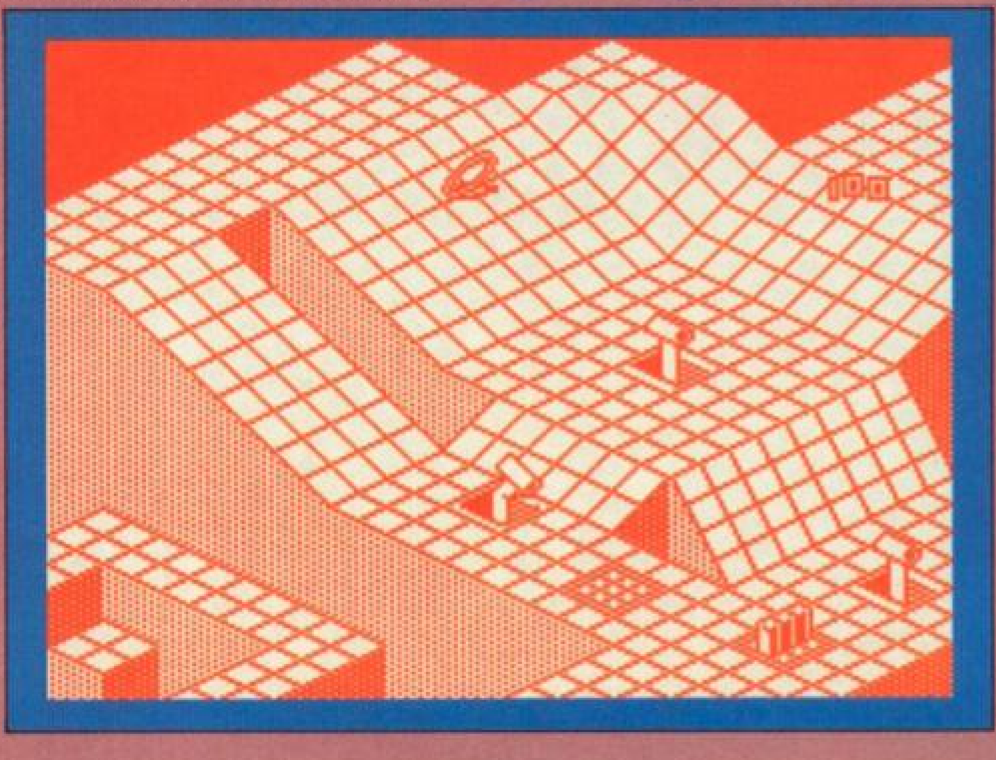
The timing system works in a similar way to the original, you start with a certain time limit in which to get to the bottom of the first screen. If you get to the bottom of the screen before time then your remaining amount gets added to your time for the next screen.

Scoring is assessed on how much time you have left on your clock when you reach the bottom of the screen, and any bonuses that you have picked up on your trip down. Otherwise the game is very similar to Marble Madness in layout and all the fine detail, like the broken ball being brushed up, is all there.

The construction part of the game lets you edit the screens as much as memory will allow. The editing system is remarkably simple, and fully icon driven with a two-thirds scale representation of the screen that you are working on with the rest of the screen taken up with the icons and chunks of isometric building blocks to manipulate. So you can move your joystick around and edit any screens that you want, to make them harder (!), easier, or just different. You can also test screens, and save/load the entire game to tape.

This selection of the program makes the whole game that much more interesting, long after Commodore 64 owners will have completed the original game, Spectrum owners will be designing better and more fiendish screens for one another. The best implementation that could have been made of this well thought out game.





CROSS WIRES

Advice from Ray Elder on protecting programs and disabling the break key in this month's technical helpline.

Program Protection

Q *Dear Sir,*
I am very fond of making games which compel the player to do some thinking to succeed. But I have one problem, I already know how to stop my programs from being broken into while loading or during the game, but I do not know how to overcome the problem of breaking in while INPUTing variables. i.e. 10 INPUT a

All the user has to do is to type in a whole lot of a's or any other letter and the program will stop with a "2:Variable not found" report and thus leaving the program open to hackers.

Please could you help me overcome this problem.
Yours sincerely
Peter Harrison, Harare, Zimbabwe.

A Well the most simple solution is to use a temporary string input instead of a numerical variable and to validate it before converting it to numerical. Suggested lines are:

```
10 INPUT LINE a$: IF a$= "" THEN GO
TO 10
20 LET flag=1: FOR I=1 TO LEN a$: IF
a$(I)<"0" OR a$(I)>"9"THEN LET flag=0
30 NEXT I: IF NOT flag THEN GO TO 10
40 LET a=VAL a$
```

This is still vulnerable, and of course the POKEs that are usually used to cause the Spectrum to crash have to be turned off to prevent the computer from locking up.

A solution would be to use INKEY$ which doesn't effect the 'protection' that causes a crash on using the INPUT lines of the screen, in this case try using the following code. Note that the screen postion 0,0: can be any position of your choice.

```
10 LET a$=""
12 LET t$=INKEY$: IF t$="" THEN GO TO
12
13 IF CODE t$=13 THEN GO TO 40
14 IF t$ < "0" OR t$ > "9" THEN GO TO
12
15 LET a$=a$+t$
20 PRINT AT 0,0;a$
30 IF INKEY$<> "" THEN GO TO 30
40 LET a=VAL a$
```

Bad Breaks

Q *Dear Sir,*
You have often said that it is possible to disable the break key, how about telling us (relative) newcomers how to do this. My brother bet me he could crash any of my programs and this would be a useful way of eliminating one way of him doing so.
Robert Giles

A The following little program should do the trick, LOAD it and RUN it and one problem solved: Note the program SAVEs the code and this needs to be loaded into your program using a line such as CLEAR address−1: LOAD "" CODE address: RANDOMIZE USR address

```
10 DATA 33,15,0,9,34,176,92,235,42,
61,92,115,35,114,201
15 DATA 58,58,92,60,40,2,254,9,202,3,
19,33,68,92,203,126
20 DATA 40,11,58,71,92,6,119,42,69,92,
34,66,92,33,0,0
25 DATA 124,50,113,92,34,11,92,42,176,
92,22,942,66,92,
30 DATA 195,158,27
40 CLEAR 65399:FOR I=65400 TO
65463: READ a: POKE I,a: NEXT I
45 SAVE "BREAK" CODE 65400,64
50 RANDOMIZE USR 65400
```

Hints & Tips

Finally, a couple of useful hints and tips from readers.

Dear Sir,
Many thanks to Toni Baker for explaining why the conditional operators would not work on the Spectrum (Machine Code Calculator Pt3. Sept). I had no idea that the code should also be in the 'B' register.

However I did manage to get around this at the time by substituting instructions which are independent of the B register, these are:

LE (<=)	*03 Subtract*	*36 LTO*	*30 NOT*
GE (>=)	*03 Subtract*	*37 GTO*	*30 NOT*
NE (<>)	*03 Subtract*	*30 NOT*	*30 NOT*
GT (>)	*03 Subtract*	*36 LTO*	
LT (<)	*03 Subtract*	*37 GTO*	
EQ (=)	*03 Subtract*	*30 NOT*	

There are two other instructions which you may find useful and they both need a single value on the calculator stack, they are:

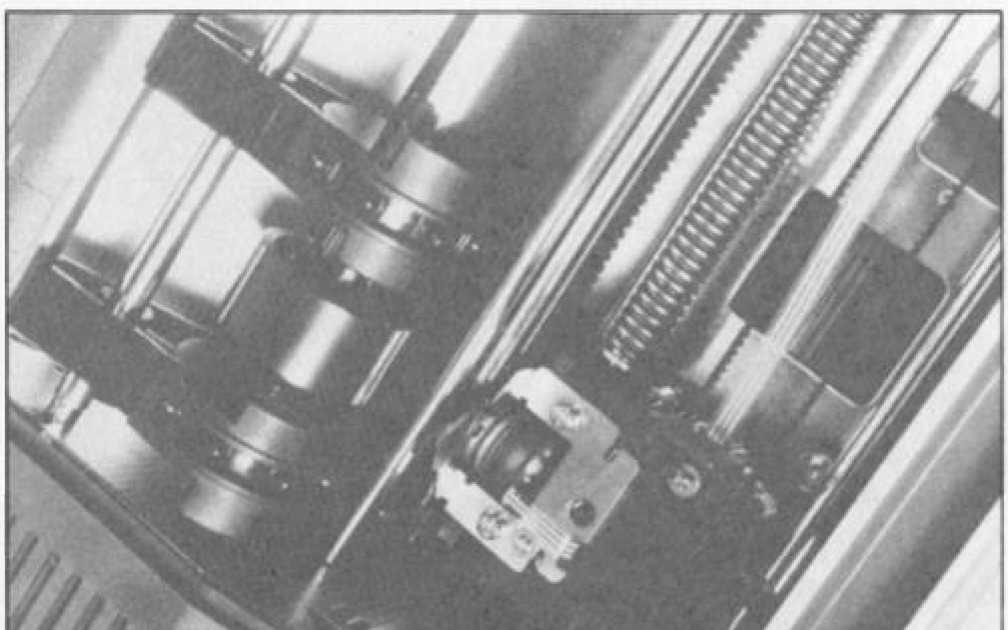
GE0 (>=0)	*36 LTO*	*30 NOT*
LE0 (<=0)	*37 GTO*	*30 NOT*

Using these saves a byte or two and you do not have to leave the calculator mode.
Yours faithfully
Ray Reeves, Harlow.

Dear Sir,
I recently purchased a Star Gemini 10xi printer and ZX LPRINT 3 to use with Tasword 3.

Just as other readers, I found that the printer control characters ruined the right justification. I have corrected this by incorporating code 32 in the control code sequences when customising Tasword. This code makes the printer print a space whilst effecting whatever change in printer mode you require.

I hope this will help other readers.
P.F. Green, Rotterdam.



SHORTCUTS

The page where small is beautiful!

When we printed B. J. Kamphuis' HEX/DEC converter in the October issue I commented on the fact that it only handled numbers 0 to 255 and suggested that you may like to extend it to cover the usual range 0-65535. Several readers rose to the challenge, but by far the most original is this effort from Mr. A. Welsh.

HEX/DEC

This program uses a trick with the RANDOMIZE statement. If the command RANDOMIZE 32678 is entered then the two byte equivalent is stored in the system variables called SEED at addresses 23670 and 23671.

If we then PEEK them and feed them into the original routine one after the other then the conversion is achieved. Also included are the two values of these addresses in decimal form and these are printed in the machine code structure LOW byte/HIGH byte, you could of course rearrange this if you prefer.

However, all good ideas usually have at least one drawback and the main one with using this system is that RANDOMIZE 0 causes the number to be placed in SEED to be completely random, although we probably all know that 0 in decimal is 0000 in HEX 50, just for cosmetic purposes the program also includes a line to trap input of 0 and to deal with it separately.

```
   1 BORDER 1: PAPER 1: INK 7: C
LS
  1Ø DATA 58,Ø,25Ø,245,6,4,2Ø3,6
3,16,252,2Ø5,28,25Ø,5Ø,Ø,25Ø,241
,23Ø,15,2Ø5,28,25Ø,5Ø,1,25Ø,2Ø1,
198,48,254,58,216,198,7,2Ø1
  2Ø FOR f=64ØØ2 TO 64Ø35: READ
a: POKE f,a: NEXT f
 11Ø PRINT : INPUT "Dec Value ?
";e
 112 IF e=Ø THEN  PRINT e;: PRIN
T TAB 7;" HEX.=ØØØØ";: PRINT TAB
 2Ø;"DEC.=Ø": GO TO 11Ø
 115 RANDOMIZE e
 12Ø POKE 64ØØØ,PEEK 23671
 13Ø RANDOMIZE USR 64ØØ2
 14Ø RANDOMIZE e
 16Ø LET a=PEEK 64ØØØ: LET b=PEE
K 64ØØ1: POKE 64ØØØ,PEEK 2367Ø:
RANDOMIZE USR 64ØØ2: LET c=PEEK
64ØØØ: LET d=PEEK 64ØØ1
 17Ø RANDOMIZE e
 2ØØ PRINT e;: PRINT TAB 7;" HEX
.=";CHR$ a;CHR$ b;CHR$ c;CHR$ d;
 2Ø5 IF e<=255 THEN  PRINT TAB 2
Ø;"DEC.=";PEEK 2367Ø: GO TO 11Ø
 21Ø PRINT TAB 2Ø;"DEC.=";PEEK 2
367Ø;CHR$ 44;PEEK 23671
 3ØØ GO TO 11Ø
```

STAR CUT

Font

This routine is yet another variation on the 'thicker' character set fonts that you all seem so keen on producing, we include this one because it uses a different approach to that of most other programs of this type and is interesting to compare with the more usual approach.

This one was supplied by Peter Zoetway of the Netherlands.

```
   1 REM Short 2
  1Ø INPUT "Address for storage:
";a: LET b=15616: FOR f=a TO a+7
68: POKE 16384,PEEK b: FOR g=Ø T
O 7: IF POINT (g,175) THEN  LET
g=g+1: PLOT g,175
  2Ø NEXT g: POKE f,PEEK 16384:
LET b=b+1: NEXT f: POKE 236Ø6,a-
256*INT (a/256): POKE 236Ø7,INT
(a/256)-1: CLEAR a-1: FOR f=32 T
O 128: PRINT CHR$ f;: NEXT f
```

Big copy

Last month we published a Giant Copy program, this month we present a copy routine that is not so large but, due to the shading effect produces a printout which would grace most computer room walls.

The program works with the usual ZX type of printer, Sinclair, Alphacom, TS2048 or GP50s and prints in four sections which have to be glued together.

Our thanks to Jean-Pierre Overbeek, yet another reader from Holland.

```
  1 REM Short 5
  1Ø CLEAR 49999: LET adr=5ØØØØ
  2Ø LET a=1Ø: LET b=11: LET c=1
2: LET d=13: LET e=14: LET f=15
  3Ø FOR q=1ØØ TO 2ØØ STEP 1Ø: R
EAD a$,tot
  4Ø LET w=16*VAL a$(1)+VAL a$(2
)
  5Ø POKE adr,w: LET adr=adr+1:
LET tot=tot-w
  6Ø LET a$=a$(3 TO ): IF a$<>""
 THEN  GO TO 4Ø
  7Ø IF tot<>Ø THEN  PRINT "Erro
r in line ";q: STOP
  8Ø NEXT q
  9Ø LOAD ""SCREEN$ : RANDOMIZE
USR 5ØØØØ: GO TO 9Ø
 1ØØ DATA "3eØØ3259c53a57c53258c
53a58c54f3a59c547c5f5cd89c4f1c13
eØ8325dc5e57e325cc52f773a59c5473
a58c54fc5f5cd4b",6155
 11Ø DATA "c4f1c17ee6Ø7325ac57ee
638cb3fcb3fcb3f325bc5c53a5cc54fØ
6Ø8cb4Øcce7c3cb79cdØac4cb211Øf2c
179c6Ø83258c5e1",6721
 12Ø DATA "233a5dc53d325dc5c26fc
33a59c53c3259c5fecØc255c3Ø6Øaf5c
5cdcdØec1f11Øf73a57c5c64Ø3257c5f
eØØc25Øc3cdc8c4",6785
 13Ø DATA "c9f5c53a5ec53cfe21caf
dc3fea1caØ2c4325ec5c1f1c93e81c3f
7c3cdcdØe3eØ1c3f7c33a5bc5ca13c43
a5ac5c5e5dd2169",7625
 14Ø DATA "c447feØØca26c411Ø4ØØd
d191Øfc21ff5a3a5ec5Ø6ØØ4fØ9Ø6Ø47
ecb27cb27cb27cb27dd56ØØb277dd231
12ØØØ191Øeae1c1",5Ø19
 15Ø DATA "c9cb39cb39cb3978cb3fc
b3fcb3f21ØØ581120ØØ47feØØca67c41
91Øfd89c9ØfØfØfØfØaØ5ØaØ5ØcØ1Ø4Ø
3Ø1Ø4Ø2ØØØØ8Ø2Ø8",3542
 16Ø DATA "ØØØØØØØ2ØØØØØØ4ØØØØØØØ
ØØØØØ21ØØ4Ødd21b8c4cb39cb39cb391
6ØØ5919780Ø6Ø8cb7fc2a9c4dd23dd23c
3b4c4dd56ØØdd23",433Ø
 17Ø DATA "dd5eØØ19dd2317 1Øe6c91
ØØØØØØØØØØØØØ4ØØØ2ØØ4ØØØ2ØØØ1ØØ1
6ØØØ1ØØ1Ø21ØØ4Øcb41cadac4cb3ec3d
cc4cb2623Øb78b1",3817
 18Ø DATA "c2dØc415c2cac4Ø6ØØ21f
f5778cb3fcb3f77231Øf721ØØ5936472
31Øfb21ØØ5a78cb3fcb3f7723Ø478feØ
Øc2ØØc521ff5836",5338
 19Ø DATA "ØØdd2137c521ØØ59Øe2Ød
d56ØØdd23Ø6Ø8cb7aca2ac5367823cb2
21Øf4Ød79feØØc21cc5c97744ØØØØ544
4ØØØØ5654ØØØØ54",4324
 2ØØ DATA "2a252e572a2524ØØØ22aa
4ØØØ228a4ØØØ3a8a4",1Ø76
```



Scroller

Another one from Holland, this time from Paul van Duijn. This is a versatile screen scrolling program which operates on the top seven lines of the screen. The program is located at address 30000 and before use you need to set up exactly the way in which it operates by using the following POKES:

POKE 30021, length_of_the_line (max 32 characters)
POKE 300001,8*PEEK 30021
POKE 30002,256—PEEK 30021
POKE 30013,start_position_of_line (0=0,0 and 255=7,31)

Paul suggests, and I found it to be true, that the best way is to completely fill the screen lines 0 to 7 with some characters and then experiment by poking the above addresses with various numbers before calling the routine with **LET l=USR 30000**

```
  1 REM Short 3
  5 CLEAR 29999
  1Ø FOR k=3ØØØØ TO 3ØØ33: READ
d: POKE k,d: NEXT k
  2Ø DATA 62,Ø,8,2Ø5,6Ø,117,8,61
  3Ø DATA 183,32,247,2Ø1,33,Ø,64
,17
  4Ø DATA 224,Ø,Ø,Ø,6,32,2Ø3,3Ø
  5Ø DATA 35,16,251,25,62,72,188
,2ØØ
  6Ø DATA 24,24Ø
  7Ø PRINT "ZX COMPUTING ZX COMP
UTING ZX COM1234567890ABCDEFGHIJ
KLMNOPQRSTUV"
 1ØØ PAUSE 1ØØ
 11Ø LET START=USR 3ØØØØ
 111
 12Ø REM POKE 3ØØØ1,NUMBER_OF_TI
MES_TO SCROLL_BY_ONE_PIXEL_TO_TH
E_RIGHT (256=COMPLETE LINE WIPE)
 121
 13Ø REM POKE 3ØØ13,START_OF_LIN
E    (Ø=Ø,Ø AND 255=7,31)
 131
 14Ø REM POKE 3ØØ16,256-PEEK 3ØØ
21
 141
 15Ø REM POKE 3ØØ21,TOTAL_NO_OF_
CHARS    (32 MAX)
```

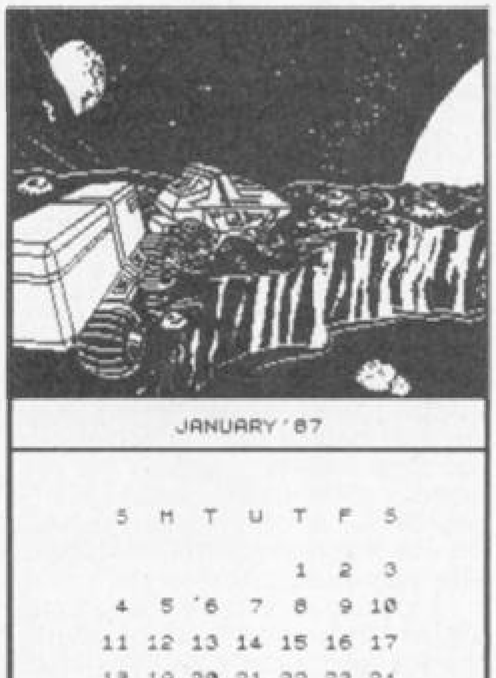
Calendar

With the New Year upon us this program from old hand Robert Glavis is most appropriate, it produces a most unique calendar made up of SCREENS, either your own designs or using those of commercial programs.

As there are twelve months in the year you will need twelve screen$ pictures, PLUS an extra one for the title page telling you it's a calendar or wishing Happy New Year etc. The program can be used for years from 1987 to 1995.

A good idea is to get all thirteen pictures on one tape then simply start and stop the tape each time the program requests and searches for the next picture.

Many Happy New Years . . .




```
  1 REM SHORT 4
  1Ø DATA 5,6.1,1,2,3,4.1,6,7,1
  2Ø DATA "JANUARY",31,"FEBRUARY
",28+LEAP,"MARCH",31,"APRIL",3Ø,
"MAY",31,"JUNE",3Ø,"JULY",31,"AU
GUST",31,"SEPTEMBER",3Ø,"OCTOBER
",31,"NOVEMBER",3Ø,"DECEMBER",31
  3Ø INPUT "WHICH YEAR? ";Y: IF
Y<1987 OR Y>1995 THEN  GO TO 3Ø
  4Ø RESTORE 1Ø: FOR F=1 TO (Y-1
986): READ X: NEXT F: LET LEAP=Ø
: IF X<>INT X THEN  LET LEAP=1:
LET X=INT X
  5Ø CLS : PRINT "LOADING FRONT
PICTURE-START TAPE"
  6Ø LOAD ""SCREEN$ : COPY : LPR
INT ''''
  7Ø RESTORE 2Ø: FOR F=1 TO 12:
READ M$,M: LET M$=M$+" '"+(STR$ Y
)(3 TO 4): LPRINT ''''
  8Ø CLS : PRINT "LOADING ";M$;"
 PICTURE"
  9Ø LOAD ""SCREEN$ : PLOT Ø,Ø:
DRAW Ø,175: DRAW 255,Ø: DRAW Ø,-
175: COPY
 1ØØ CLS : PRINT AT 1,16-(LEN M$
/2);M$''''''"          S  M  T  W  T
  F  S"
 11Ø LET PX=9: LET PY=(X*3)+4: F
OR G=1 TO M: PRINT AT PX,PY-(1 A
ND G>9);G: LET PY=PY+3: IF PY>25
 THEN  LET PY=7: LET PX=PX+2
 12Ø NEXT G: LET X=INT (PY-4)/3
 13Ø PLOT Ø,153: DRAW 255,Ø: PLO
T Ø,175: DRAW Ø,-175: DRAW 255,Ø
: DRAW Ø,175
 14Ø COPY : NEXT F
```

FIRST STEPS IN MACHINE CODE

Do you want to learn Machine Code? ZX regulars tell how they set about mastering M/C

Ray Elder

I first experienced the joys of computing with a £30 Atari VCS cartridge which gave 62 BYTES of programmable memory, just enough to write a simple workable quadratic equation solving program in a strange mnemonic type of code.

The next step was to purchase a secondhand ZX81 including 16K rampack and a couple of books for £70. I well remember the joy of writing the simple soundless, black and white block graphics wonders of that era, and also opening the *"Mastering Machine Code on your ZX81"* book by Toni Baker.

I also remember closing it after being completely bamboozled by the first three pages — Hex numbers and memory locations!

After about four months of mastering the easy but intriguing instruction manual (remember LET EGGS=12 and finding the square root of an EGG?), I was forced by a rampant Pancreas to spend a few weeks in hospital. My wife brought in all my computer books, I was well and truly hooked by then, and the only one I hadn't yet managed to read, due to memories of my first abortive attempt, was the Toni Baker one.

Sheer boredom drove me to open it again and, with my slightly greater experience, I found some areas actually made a strange kind of sense...

I actually read the book from cover to cover twice and in the process found a new awareness and excitement developing.

When I was discharged I hurried to try out this new knowledge by entering the first program.

It didn't work.

However I was now determined to succeed and corrected the misprints and tried again, this time success.

I purchased Ian Logan's book on ZX81 machine code and my knowledge deepened, I read all the articles I could find but most importantly I wrote, rewrote and rewrote again code until it worked, my biggest effort being a cricket game of some 4K of machine code all hand assembled (I couldn't afford an assembler).

So, advice to anyone embarking on this venture. Get a good grounding in BASIC first especially of the operations of PEEK and POKE. Buy a book which is written in the style which you find readable, Toni Baker's *"Mastering Machine Code on Your Spectrum"* is the equivalent to my first book but it takes some slow careful reading. Ian Logan's book is also good but check out a good bookshop with a variety of titles for one which suits you.

Persevere. Buy an assembler, they'll save you much time and frustration, I use the Picturesque one, it suits my needs, and start small, convert small routines from a larger program first and build it up into a collection of routines which can be called from one master routine.

If you have developed your programming on a modular or structured basis then the transition to machine code will be an easier one to make. I was a notorious spaghetti programmer and I suffered until I learnt to plan and simplify.

I wish you well...

Clyde Bish

There are, I think, two approaches to learning M/C. I've tried both and it will be obvious from what follows which was best for me.

You must appreciate that Z80 Assembler is as much a language as Basic, Pascal, French and German. Thinking back to learning one of the latter, I didn't learn isolated words out of context, but learnt to use them in phrases. I applied the same logic to Assembler.

I looked at simple routines (rather than learnt lists of op-codes and what each would do) via books that explained what was happening, e.g. Hewson & Hardman's *"40 Best Machine Code Routines for the Spectrum"* (Hewson Consultants), Webb D. *"Supercharge Your Spectrum"* (Melbourne House) and S. Webb's *"Practical Spectrum M/C Programming"* (Virgin) (which puts routines into context via a game and demonstrates that there is no such thing as a big program. Only a collection of little ones!) I still use the first title mentioned as a reminder of what certain op-codes will do.

I also disassemble routines (e.g. MCODER III) to see how other writers have gone about it.

It is essential to have an assembler. Working with just the decimal or hex equivalents is useless.

Most of all it is essential to have a good working knowledge of the Spectrum firmwear (D_FILE, System Variables etc) as all routines operate on these. A ramble through the ROM is useful, picking up useful routines, afterall why rewrite ones that are already there?

Don't try for the sky too early. Start with short routines to test out instructions like the JR condition, displ. ones., until you're familiar with what they can do. Remember that what is in BC returns to the screen so you can arrange for this to hold a suitable figure to tell you whether you've jumped or not.

Finally, remember that M/C is not the be all and end all of programming. Many v. good programs only use M/C where necessary, e.g. "The Forest", "Tasword" and the recent super series by Alan Davis.

Carol Brooksbank

It should be 'How I *started* to learn machine code,' because I don't think I shall ever know all there is to know about it, and I am certainly still learning now.

About three years ago it became clear that a lot of what I wanted to do with the Spectrum was impossible in BASIC, so I looked for a book about machine code. Toni Baker's *"Mastering Machine Code on your ZX Spectrum"*

(Interface), had just been published and that was the one I bought. It is written in a style the beginner can understand, and yet goes pretty deeply into the subject. Toni's printers had made quite a lot of mistakes in the early copies, and an errata sheet was produced. I decided to see how far I could get before I had to send for it. In fact, I never did send for it. I soon discovered that the listings with the mistakes were the ones which really made me study. You cannot get away with typing in the listing and going on to the next chapter without bothering to understand what you are doing if you have to trace a printing error and put it right before the program will work. You really have to work and grasp the logic and the procedures of the program.

Toni's book is still the most important in my library. I have lost count of the number of times I have been stuck over something in one of my own programs, and have found some hint or explanation in 'Mastering machine code' which has shown the way forward. Once I started to write programs, Ian Logan's *"Understanding Your Spectrum"* and *"The Complete Spectrum ROM Disassembly"* by Ian Logan and Frank O'Hara (both Melbourne House) became essential. William Nitschke's *"Advanced Z80 Machine Code Programming"* (Interface), took me a stage further, and I now find Rodney Zaks' *"Programming the Z80"* (Sybex) invaluable.

I have learned all I know about machne code from books, and although I now have quite a collection of books on the subject, those are the ones I would grab if there was a fire.

David Nowotnik

I found BASIC on the ZX81 was marvellous until I wanted to work at more than a plodding pace; then Z80 machine code became a necessity. My first thought was to go for one of the all embracing reference texts; I chose *"Programming the Z80"* by Rodney Zaks. That I found so heavy going, I almost gave up. Fortunately, about the same time, Toni Baker brought out her book *"Mastering Machine Code on your ZX81"*. That was my salvation. Easy to read, logically building up knowledge, with example routines to try all through the book. The Spectrum equivalent of this book is just as good.

Analysing, and discovering for yourself how other people's machine code routines work is an excellent way of learning. *"40 Best Machine Code Routines for the ZX Spectrum"* by John Hardman and Andrew Hewson is a book which builds on this principle. All 40 routines are analysed by the authors; they do all the hard work for you!

If you are not a member of a local user group, then join one! Most groups will have a wide range of abilities amongst their members. You'll find beginners like yourself; I found talking over machine code problems with a group can be very helpful (and it's particularly good for your confidence in your own abilities when you can start to pass on advice). And, of course, in a user group there'll be experts at hand to help you as well.

Incidentally, I now find the Zaks books a very useful reference text.

And that's how I learnt machine code!



Toni Baker

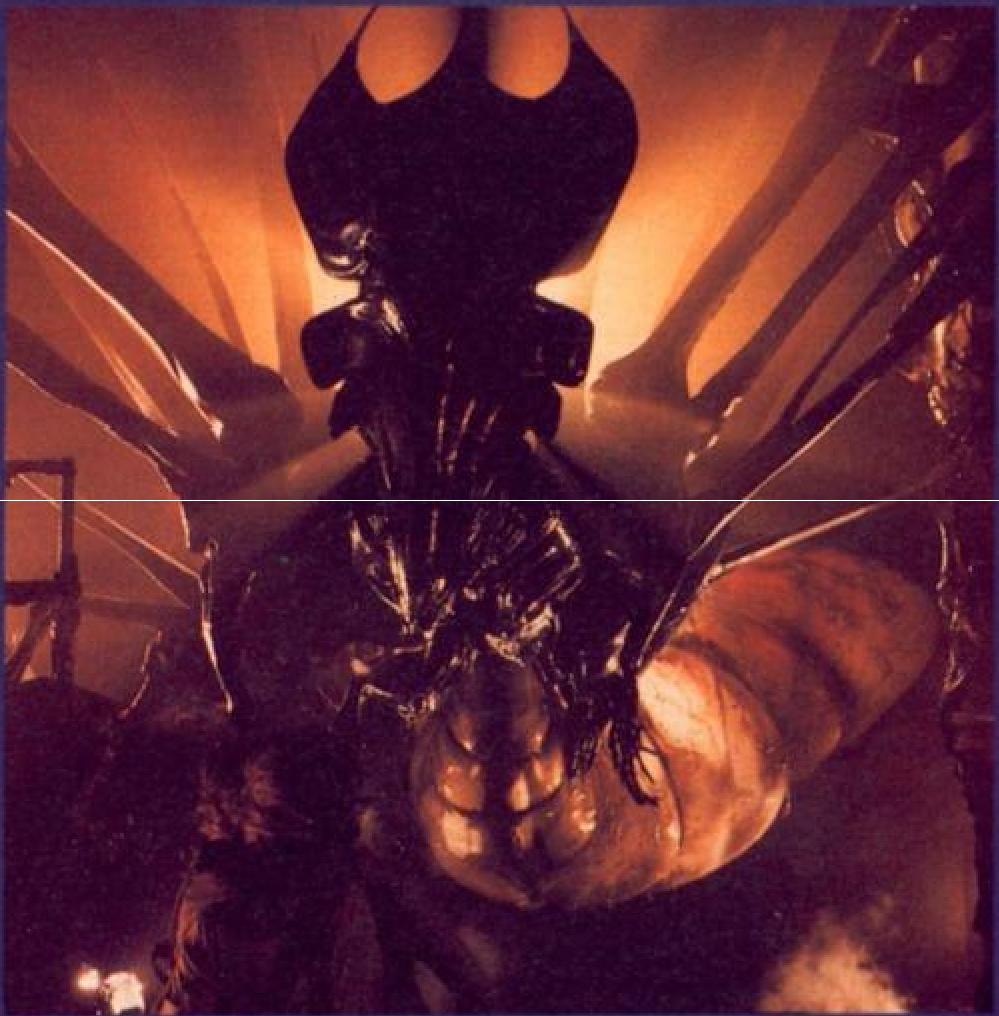
Like many other people, my first ever experience of home computers was the good old Sinclair ZX80. This, in case you've forgotten what it was (or never knew in the first place), was a white plastic doorwedge-shaped slither about six inches square with a tiny keyboard printed on its surface. It had only 1K of RAM altogether — although you could upgrade this to 4K by plugging a little white box into the back. The ROM, believe it or not, fitted into 4K. The ZX80 knew nothing about integers greater than 32677, and decimals were a complete mystery to it. If you can't remember the old ZX80 take a trip to the Science Museum — you'll find one there.

The ZX80 did, however, have a redeeming feature. Three keywords which I didn't understand. Nobody I spoke to knew what they were or what they did. The words were PEEK, POKE and USR. PEEK and POKE seemed to do nothing except cause abject confusion, and the purpose of USR seemed to be to invariably crash the machine. So I got hold of a copy of Rodney Zak's book on the Z80 chip. Before long I worked out that if you POKEd into consecutive addresses the codes for Z80 instructions, and then did USR of the first address — the machine didn't crash!

Before long I was finding out the best places to put machine code — I'd got it all sussed out — and then DISASTER! The ZX81 came out and I had to learn things all over again. The Spectrum came out and the same thing happened. Now at last I think I'm beginning to get the hang of the Spectrum. I use Dr Ian Logan and Dr Frank O'Hara's book (The Cpmplete Spectrum ROM Disassembly) like Jehovah's Witnesses use the Bible. And the rest, as the saying goes, is mystery.

ALIENS

Head biting, gut ripping, face hugging, people shredding . . .



Aliens
Electric Dreams
£7.95

Ripley, the sole survivor of the crew of the Nostromo that was chomped by a single Alien in the original film, now reluctantly leads an assault on a base packed with alien eggs, warrior, face huggers and queens. Their mission is genocide!

The target is a human colony that was built on the remote planet containing the alien eggs. The Aliens wiped them out, now they're after you . . .

You command a team of six troopers led by Ripley which includes Gorman, a space marine, Hicks who is noted for fast reactions, the android Bishop, the tough Vasquez and Burke (the Company man). Each is armed with an alien-frying smartgun that can take out a warrior with a single shot at its head (or three to its body) and is also useful for blasting doors open or sealing them by taking out the lock mechanism.

Each crew member must be ordered separately either by issuing commands (e.g. 5N , move north five rooms) or by direct joystick control. I found this to be the better, although slower, method as you could ferry your crew room by room through the deadly maze of corridors. If you find an alien (or an alien finds you) you'd better be quick before its jaws get you. You're given a fighting chance as a proximity alarm warns of a nearby alien.

If a crew member is attacked while you're controlling someone else then the aliens will try to take them over. This is represented by a health bar, which is normally green, turning yellow. You can save them if you can get to them and kill the alien before the bar turns purple. Panicking at this stage can cost you your entire crew!

Acid

Aliens can still be deadly even once you've killed them as they can leave impassable pools of acid in front of important doors and bio-mechanical growth all over the walls. This must be cleared otherwise it will spread with disastrous effects. The air ducts will become infested with alien eggs and soon you'll be swamped with face huggers. Besides if it takes control of the generators you'll be plunged into darkness and a hopeless situation.

The screen display shows the view of one of your crew as well as their gun sight. Underneath that is a picture of that team member, their current ammunition level and a display for each human showing their current state of health and the number of the room they're in. These numbers are essential to find your way through the complex. I also found it useful to blast the doors and the locks to mark a route through the maze for the rest of the team to follow, since you haven't got time to ask directions when you're being chased by aliens.

Some rooms have special significance, such as the armoury which automatically recharges your smart guns, the control and generator rooms that must be defended and the Queen's chamber that must be taken to win the game.

This is not an easy game to play as it demands almost total concentration coupled with a steady fire button finger and a cool nerve. One slip could cost you your entire mission.

Very few games have such atmosphere that compel you to return for more and more until you finally succeed. Aliens is an excellent game based on a superb film and is undoubtedly the best licensed game yet produced.

MONSTER HIT



RIPLEY
001
GORMAN
007
HICKS
001
GORMAN
00000000
BISHOP
001
VASQUEZ
001
BURKE
001



RIPLEY
001
GORMAN
001
HICKS
007
00000100
BISHOP
009
VASQUEZ
001
BURKE
001

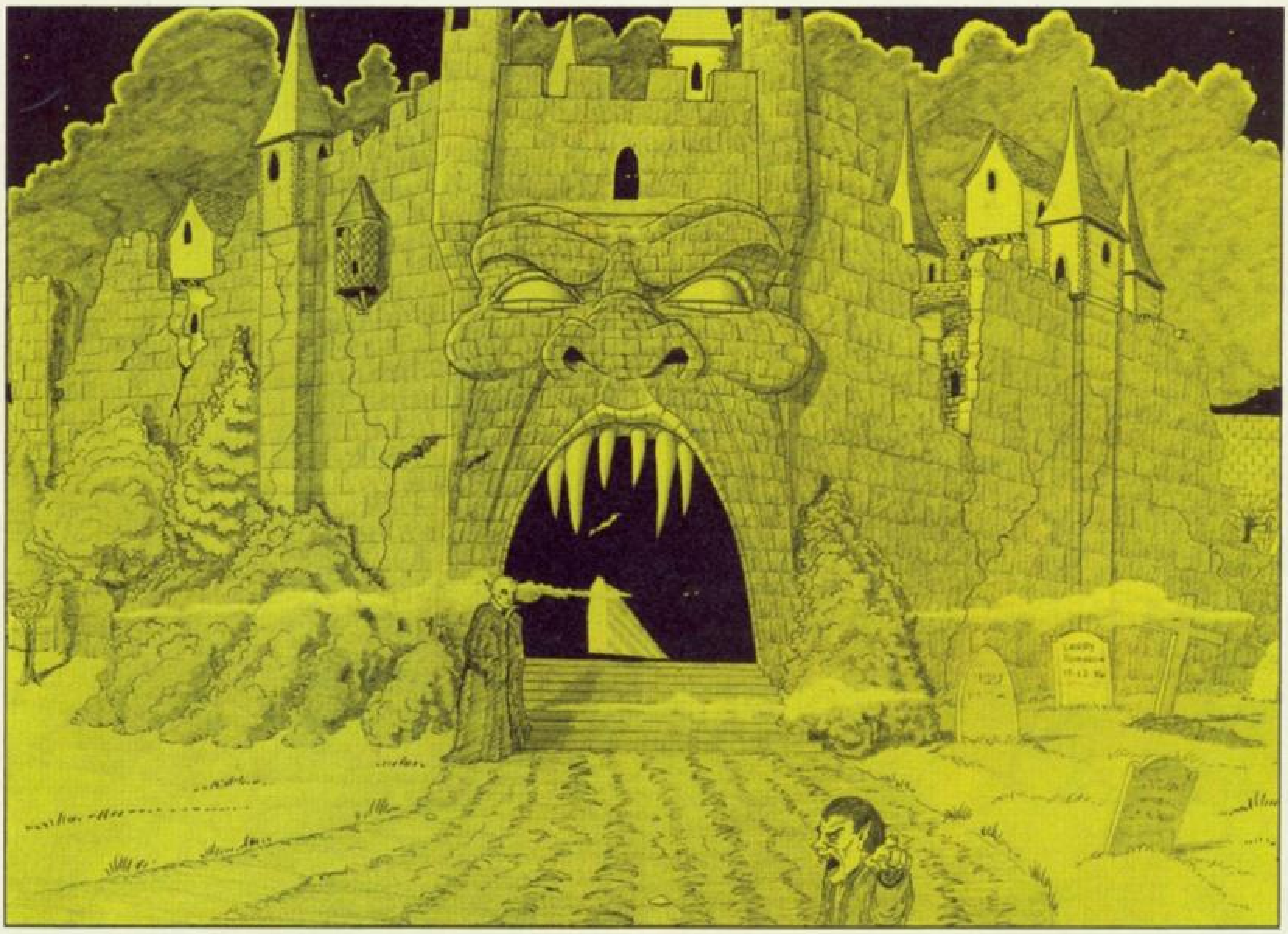


RIPLEY
001
GORMAN

HICKS
010
HICKS
00000300
BISHOP
001
VASQUEZ
068
BURKE
001







MINDPLAY

Very short introduction this month because there's a lot to fit in. You see, at the time of writing, the Christmas rush is in full swing, so lots of games have come in for review. Four major ones — all of them high quality — are reviewed in depth here. Among them, Dracula is the first game ever to be given a rating by the British Board of Film Classification. CRL obtained this as much for publicity as concern for the country's youth I would think. Very young children might find it distressing, but I doubt they'd understand the language. There's certainly nothing corrupting to, say, over twelves; but such is the stupidity of Britain's rating system, if a something is deemed unsuitable for three year olds, it's banned from fourteen year olds as well.

Also The Colour of Magic, a licensing deal which would seem to have been motivated by the suitability of the story for a game rather than to cash in on the title (Terry Pratchett isn't exactly well known, is he?) Now that's something Ocean would never dream of . . .

But enough of these deranged ramblings, on with my . . .

Picks of '86

1986. A good year for Spectrum adventuring? Now we're into '87 that question bears looking at.

It was the year in which the Quill became perfectly acceptable for writing full priced adventures with, thanks to the astonishing chart success of Delta 4's games. A new adventure utility, Incentive's Graphic Adventure Creator, arrived to rave reviews. It has so far failed to spawn the number of games that The Quill has been doing, and looks set to be knocked off its temporary pedestal by Gilsoft's retaliation, a 'super-Quill' called the Professional Adventure Writer. Gilsoft also continued their admirable policy of improving their current Quill utilities with The Press text compressor, which contains numerous other desirable features.

CRL emerged as a major — and very successful — adventure publisher by bringing us games from Delta 4 and Saint Brides. Level 9 showed off their stunning new parser in Price of Magik and Worm In Paradise, but persisted with their abominable graphics. At the end of the year there was a flurry of licensing deals, ranging from Asimov to The Archers; previously this practice had been confined mostly to arcade games.

I did not feel, however, there was one game which really stood out, and I feel Spectrum adventures are not as good as they could be. Although the general standard from mainstream software houses is high and more consistent nowadays, the last year has lacked sparkle. Like arcade games a year or so ago, adventures have become slick

but not innovative. The licensing deals have provided some much needed originality from a market I feared was running out of ideas. Very few budget adventures have reached the high quality of some of their arcade counterparts. And still NO-ONE — to my utmost frustration — packages Spectrum adventures properly, something I'll complain about in more detail next month.

What have been the good games of '86? Well, this is my top five:

1. The Boggit (CRL, £7.95): Very silly, slick, extremely enjoyable, from Delta 4.
2. Twice Shy (Mosaic, £9.95): The most polished game of the year with many of the features I like most in adventures. Still need to read the book though.
3. HRH (8th Day, £6.95): Surprise of the year was this amusing Quilled satire from a minor budget house. A much-needed breath of fresh air with lots to do.
4. Jewels of Darkness (Rainbird, £14.95): Despite my reservations, you get many weeks of adventuring for your money.
5. Souls of Darkon (Bug Byte, £2.99): Another professional game with a great atmosphere, interesting problems but fussy vocabulary. Makes the top five because it's so cheap.

Only two other budget games are worthy of a mention; Seabase Delta which is an enjoyable adventure let down in other areas, and John Wilson's Everyday Tale, a Hobbit spoof with fun puzzles at only £1.99. Write to me for details about this or HRH.

Talking of John Wilson, I asked Rochdale's master adventurer — and much valued contributor to Mindplay — what his favourite games of the year are. His top five reads very differently to mine, not least because we have different tastes; he preferring what I call 'puzzle' games:

1. The Boggit: No difference here however. Says John: "It had to be this, for the sheer amount of pleasure it gave me. Not the best Fergus McNeil has ever written, but full of humour and steeped in atmosphere. For a good laugh, try using the words BOGGIT, BORED and DELTA 4 in part 2."
2. Terrors of Trantoss (Ariolasoft, £9.95): Never played this one myself, but RamJam (of Twice Shy and Valkyrie 17 fame) wrote it, so it must be good. According to John, "this would have ousted Boggit from the top spot bar for the BREAK bug. A different approach to adventuring, with the ability to choose one of three possible ways to carry out most acts, and some very tough puzzles. I loved the way you could select which brother to carry out an action."
3. Very Big Cave Adventure (CRL, £7.95): "A parody of the very FIRST adventure but done in a wicked way by the girls of St Brides. Some very involved puzzles only spoilt by the attempts to make the user feel as though he was not quite in control of the game."
4. Rebel Planet (Adventuresoft via US Gold, £9.95): "A vast improvement over the criminally bugged Questprobe 3 that AI released prior to this. Full of nice surprises — who would have thought that singing to a crag snapper could prove beneficial?"
5. Aftershock (Interceptor, £9.95): "Would have been higher but for the 80% price increase compared to Interceptor's last release. However, the artwork of Terry Greer nearly made up for that."

Both myself and John deemed **Bugsy** (CRL, £7.95) worthy of a mention. He feels it "a great idea spoilt by lack of puzzles," while I feel it was good fun but lacking that certain something to make it 'mega'.

Still, enough looking back. Let's hope 1987 brings some exciting products.

The Case of the Missing Helpline

"Preposterous, Holmes! Helplines don't just vanish!" bumbled the ever-affable Watson.

"That would seem to be exactly what has happened here, my dear Watson" replied the brilliant detective, still staring intently at the copy of ZX Computing. "And I know the culprit — Peter Sweasey!" Both men turned round to face the startled adventure columnist huddling in the corner.

"Very well, I admit it," stammered Sweasey. "But I can explain. Sorry dear readers. There are several reasons for its disappearance. I've had half the time I usually do to write this month's column, but twice the amount of quality releases to review. I've already used up all my space and much of my time with the reviews and picks of '86. Since it's only two weeks since I wrote the last helpline, not that many letters have reached me, and many of those which have required me to go back to check the original game, which as I've already explained I didn't have time to do. But fear not, next issue it will return. And in the meantime, all those who have written will receive personal replies as usual, so no-one has to wait too long."

"Could you help me? There are some cases I cannot solve," begun Holmes. "Like how to start Bugsy. How to avoid instant death in Kayleth. Then there's Dracula, Hunchback, The Boggit . . ."

"Stop!" cried Sweasey. "Of course I can help. Just fill in the coupon printed here and send it to Mindplay, ZX Computing, No 1 Golden Square, London W1R 3AB."

A few rules: British correspondents, please enclose a stamped, addressed envelope if you want a personal reply rather than wait some months for the magazine to come out. If you are writing from abroad, just enclose an envelope — I'll add the postage. I try to respond within two months but I can take longer (on the other hand, you might receive an immediate reply). **I only deal with adventures.** Not arcade games; nor technical adventures (Gargoyle games included, not even Heavy on The Magick). Finally, please put the name of the game you're writing about on the back of the envelope.

THE COLOUR OF MAGIC

Piranha/Delta 4
£9.95

Er, have you read The Colour of Magic? No, me neither. I want to after playing this. Unfortunately, I couldn't get it from any of my numerous local bookshops or libraries. I will order a copy, but the book should have been supplied with the game, or at least an optional, book/game package should have been available. The reason I'm getting so worked up about this is that to fully appreciate this game, I get the impression that to have read the book helps. Doing so might also make the game easier.



Rincewind is a rather useless magician, who knows only one spell, which he cannot produce. He lives on Discworld, which is carried by four gargantuan elephants on the back of a turtle. And he's broke. So he can hardly believe his luck when Twoflower, Discworld's first tourist, offers him a vast sum of money to be his guide. Rincewind is the only person who can speak Twoflower's

language, being an excellent translator. But guiding the blundering tourist — who is quickly kidnapped — is not easy. You must also cope with his luggage, which propels itself using hundreds of tiny feet and will viciously protect its master and his iconograph, which is like a camera but contains a miniature imp who paints very quickly.

This marvellously silly — and original — story is just the start. Having unsuccessfully tried to escape the city with Twoflower's advance fee, I, as Rincewind, encountered Death himself ("it had to be Death. The empty sockets were a 'dead' giveaway and the scythe over one shoulder was another clue"). Death (he has to visit wizards personally for them to die) was expecting me to be somewhere completely different. His system has been screwed up, and he's none too pleased about it. Another problem . . .

Because it's based on Terry Pratchett's book, COMB has a different style humour to Delta 4's normal unsubtle farce and spoofery. It's more gentle, and comes largely from the strangeness of Discworld and the Twoflower's naivity.

In some other respects this bears the familiar hallmarks of previous Delta 4 successes. There's loads to be read (though EXAMINE is disappointing), and many amusing little occurences, like the Iconograph complaining he's out of film because we took too many pictures at the Whore Pits. The graphics are much better than in any other Quilled game — Delta 4 don't get the acclaim they deserve in this area — and for the first time ever in Quilled product, they do not scroll up with the text.

However, I felt distressingly out of control during COM. To progress with the game, it would seem you must perform individual actions in the right place which trigger off a whole sequence of events. So you spend time wandering about looking for which action to perform . . . somehow, whilst I enjoyed the descriptions, I felt I wanted more to do.

Because of this, I don't feel COM is Delta 4's best. However, since there are four parts, there's lots to see and be entertained by; which means good value for money.



DRACULA (15)

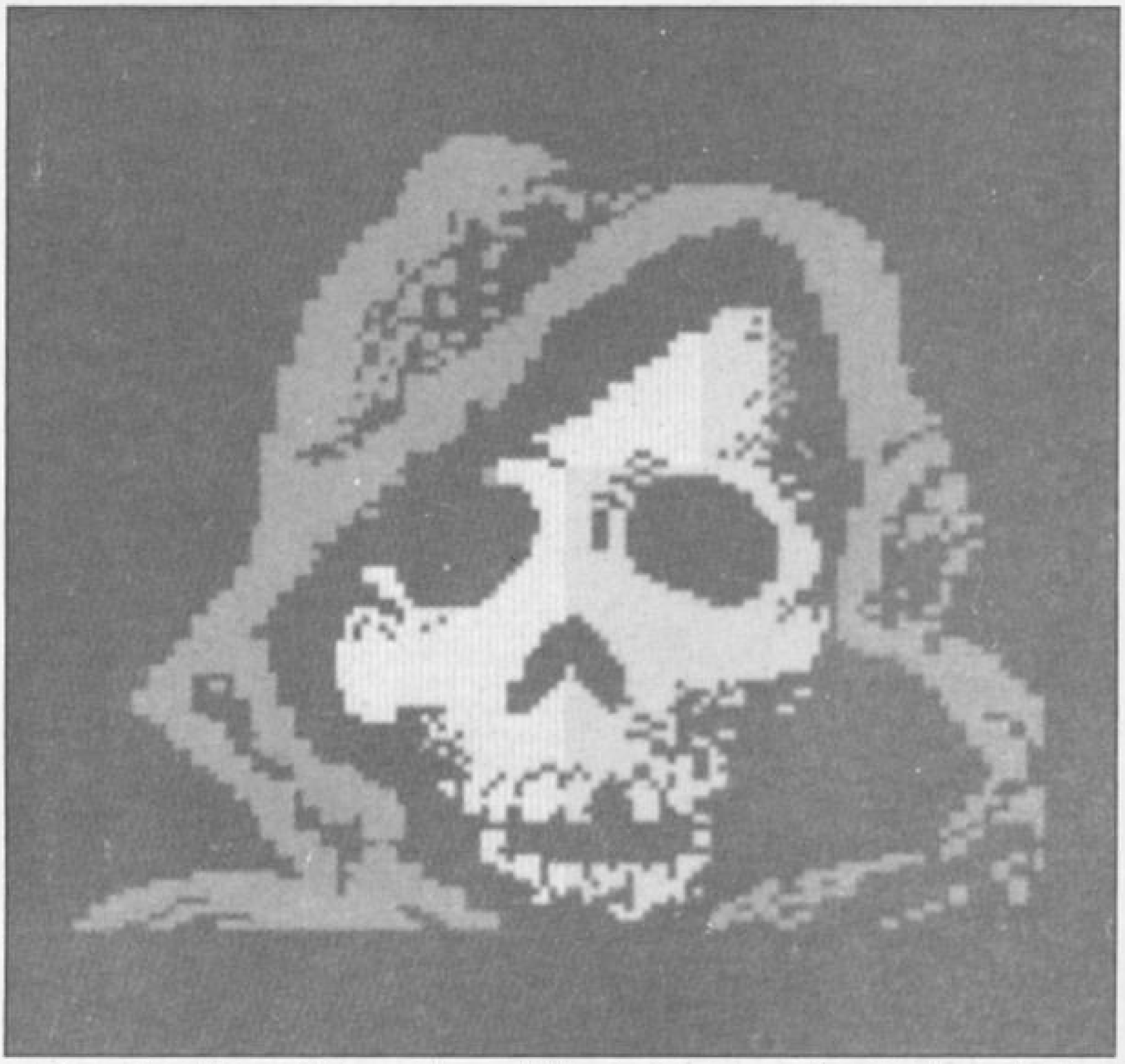
CRL
£8.95

Horror has never been treated properly in adventure games. Instead of attempting to build up atmosphere, or shock, most authors abuse the genre by making camp jokes or mixing up various legends (for example having vampires and werewolves in the same game); either that, or the game is really cliched.

How refreshing and welcome Dracula is. By taking the story seriously, and sticking closely to the original Bram Stoker novel, instead of the silly Hollywood version, this game comes closer than any I have played to being quite spine-chilling. It isn't frightening — I don't see how a series of characters on screen could be — but the horror elements can be appreciated.

There are three separate games, which can be played independently though the narrative flows through them. In 'The First Night', you are Jonathan Harker, a young solicitor visiting a client in Transylvania who has purchased a house in England, and wishes to bring with him some local soil. The game concerns your troubled night at The Golden Krone Hotel, the last stop over of your journey. 'Arrival' finds you realising that you are a prisoner at Castle Dracula, and in danger! Your task is escape. In 'The Hunt' you play a different character, Doctor Seward, a psychiatrist who receives a peculiar letter about the 'undead' from his friend John Harker. But he has problems of his own: an inmate from his asylum is behaving murderously. Little does Seward know that the normally subdued Renfield is under the influence of . . . Dracula!

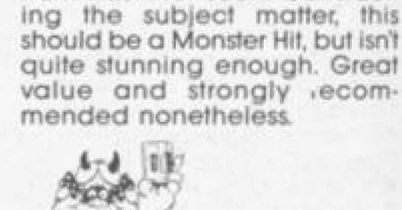
The game has reams of description, some of the longest I have seen. Evocatively written, it creates a suitable atmosphere. You anticipate in fear until, suddenly, the shock moments of terror (well, supposedly) occur. On the first night you have an apparition. ". . . he has no face!! Just raw, burned flesh!! . . . the apparition drips fetid blood onto my face — evil exudes from every pore of its vile being."



Then we have the graphics. Although the game is Quilled, CRL have wisely avoided having split screen location graphics, clumsily and slowly drawn with the Illustrator; their crudeness would make a mockery of the game's atmosphere. Instead, when a frightening event occurs, a keypress causes the screen to go black and, in the middle, a small graphic appears to illustrate the event. But what graphics! Very high resolution and quality (they look digitised), they are suitably gruesome in subject matter and genuinely effective.

The game has flaws. First is the character set which, though perhaps suitable in style, is difficult to read and entirely upper case; some of the mounting dread is lost because you are straining to read the text. The choice of colours — yellow on blue — is not very suitable. There's no ram save/load in a game where death occurs more frequently than others, and the vocabulary is occasionally too limited. Worst of all, in a game with this much text, are the instances of poor punctuation, spelling and proof-reading. "Transilvania" for goodness sake! And surely someone at CRL must know the difference between "it's" and "its".

Despite these niggles, I enjoyed Dracula a great deal. It really does feel like participating in a novel; furthermore, it must be the closest conversion fro another medium the Spectrum has ever seen. Considering the subject matter, this should be a Monster Hit, but isn't quite stunning enough. Great value and strongly recommended nonetheless.

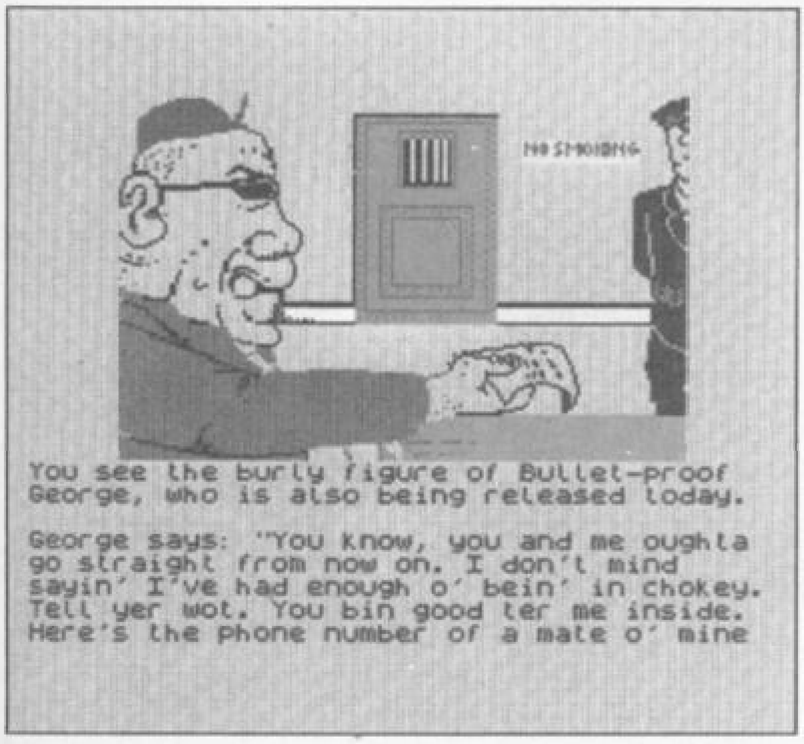


DODGY GEEZERS

Melbourne House
£7.95

So then me old china, will ver latest Lever/Jones comedy be a nice little earner for Melbourne 'ouse? You've just finished your stay at one of Her Majesty's hotels, which you were sent to because someone grassed about your part in the Long Ditton Spaghetti Caper. Now you want to pull off a big job to set you up in the Costa Brava for life; and you can have your revenge in the process. In part one of this multi-loader, you must recruit your gang from the local, East End lowlife — very colourful characters they are too. In the second part, you have to pull off the caper itself.

The game was developed on the Quill but has been 're-programmed'. In contrast to when Melbourne House have done this in the past, Dodgy Geezers does look very different to your average Quilled game. A noticeable improvement are the few but high quality, instant, cartoon graphics. However, the input routine has been badly programmed, and infuriatingly repeats letters several times if you type at speed. My copy is bugged — you cannot load from tape — which must be corrected before this is released. And the instructions say you can use IT and THEN commands: this is untrue. All not the quality expected from Melbourne House.



The Lever/Jones team gained a good reputation with their wonderful satire Hampstead, and the spoof Terromolinos. I think fans of these two games will be slightly disappointed with Dodgy Geezers. The humour is not so farcical nor obvious. The game is true satire, which means it's less funny than we've grown accustomed to. For example, the twosome point out how the pet shop is stocked with animal food made from other animals, and show us "groups of rosy faced children playing with spent fuel rods" at the nuclear waste dump. This is not a criticism — in fact it's admirable of Lever/Jones to try something different from the new well trodden silly style — but don't expect a laugh a minute. Unless, that is, you find cockney language funny. Melbourne House expect us to — "there's lots of catchy phrases for the kids to pick up" waffles the PR blurb patronisingly — but it isn't used nearly as entertainingly as the Chicago style in that other criminal comedy, Bugsy.

Time plays an important part in the game: it changes through morning, afternoon, evening, night and the days of the week. Certain locations will only be accessible at certain times — so no shopping after dark. The actual adventure has a different style to previous Lever/Jones successes. There's less to solve. Instead, a lot would seem to depend on being in the right place (not easy — the place is a labyrinth) at the right time having done the right thing earlier on.

As such, Dodgy Geezers is not particularly satisfying to play. The text doesn't liven the game up much, being somewhat sparse; and an unresponsive EXAMINE fails to enhance bare locations. Dodgy Geezers has an original plot and tries something different, but I don't think it's entirely successful.

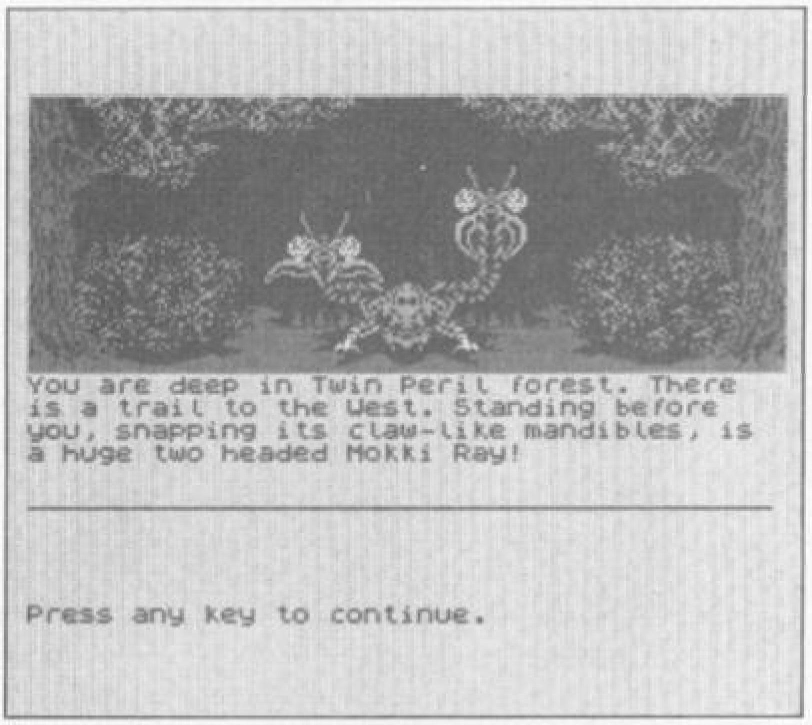
GREAT

KAYLETH

US Gold/Adventuresoft
£8.99 (48K)

Although Adventuresoft, this game's authors, are not exactly the same as the late Adventure International, the two companies are related, and Kayleth is visually similar to AI's output. What pleasantly surprised me is that Kayleth is much better than AI's games, which were always too close to Scott Adams' poor style for my liking.

The plot is good, being based on a story from Isaac Asimov's Science Fiction Magazine. The Zyroneans were an advanced, peaceful civilisation until the arrival of the awesome being Kayleth and his obsessional craving for Chromazin. Kayleth enslaves the Zyroneans using his android armies. You, a loyal Zyronean, had planned to liberate your planet when you were captured. When you regain consciousness, you're strapped to a conveyor belt, heading slowly but surely towards some menacing steel claws. You remember nothing.



Like many of AI's games, Kayleth sets you right in at the deep end with this predicament. Your method of escape, and other events early on in the game, lead you to discover something rather alarming about yourself. You are a . . . but I mustn't give the plot away!

The graphics are really high quality; AI's always were the best on the Spectrum and these carry on the tradition admirably. They are instant, high resolution, colourful and detailed. All of the numerous locations are illustrated, an impressive achievement since the graphics are not notably repetitive (although many are symmetrical, which presumably helps). An added bonus is occasional animation. Yes, the steel claws grab at you while emitting drops of liquid, the Mokki Ray's two heads move up and down, a hover-pad spins and so on. This is novel and quite well done, though sometimes too jerky. It also causes the typing speed to slow down dramatically.

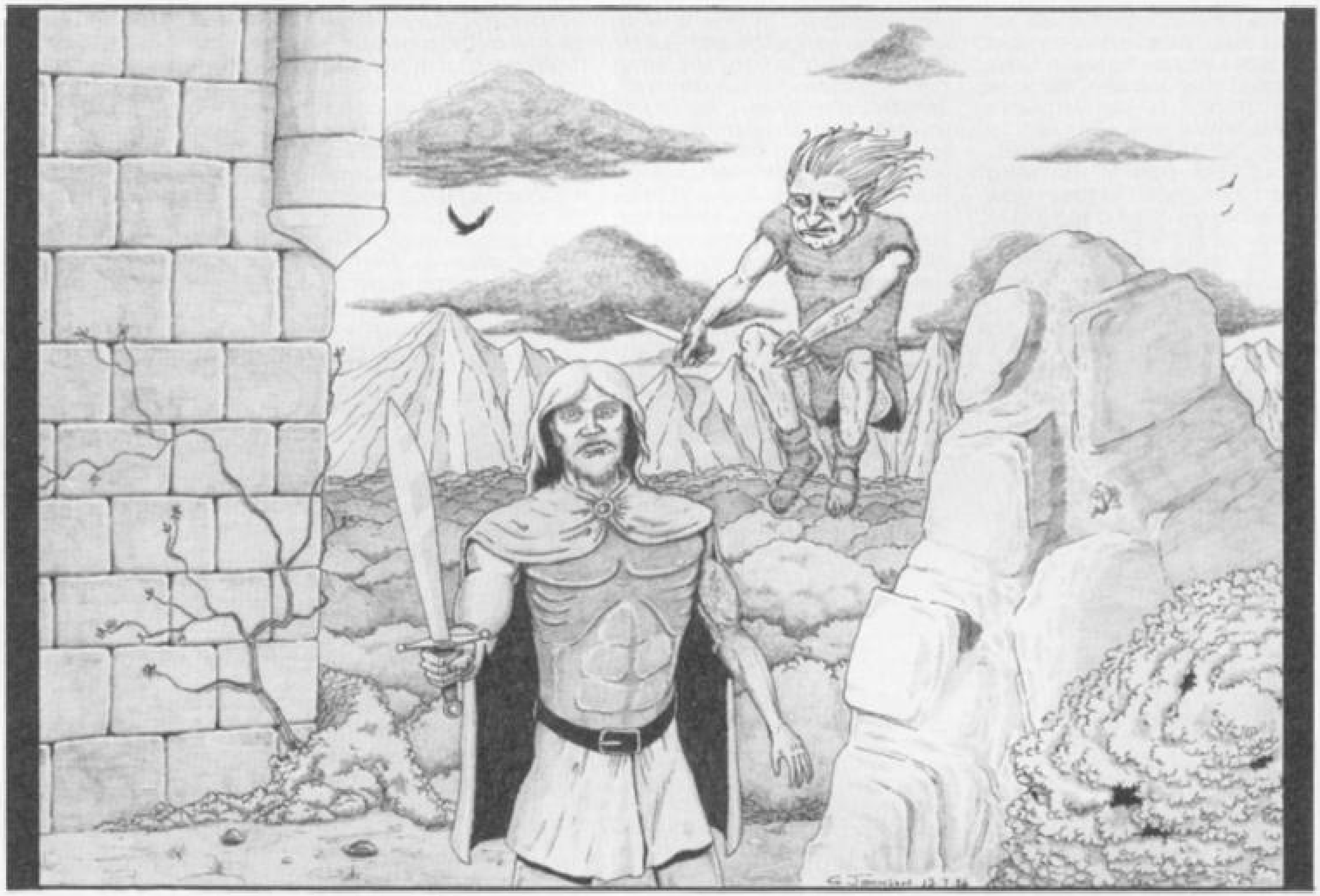
Normally, graphics of this standard mean low quality description and adventure, but I'm glad to say this is not the case in Kayleth. While text is not up to Level 9 standard, it isn't ridiculously brief either. There are pleasing touches like one of four random responses to typing HELP (they all mean no, but the variety is welcome!) Vocabulary is friendly enough, and the parser is multi-word. It has sophisticated features, including use of IT and THEM in sentences, RAM LOAD and SAVE, GET/DROP ALL, and best of all, BOM — Back One Move, a useful feature which even Level 9 didn't include in their last release.

Also notable was the Preview option given at the start of the game, which shows the player some of the locations he will encounter. Helpful if you're not good enough to reach them when playing for real!

There are some fun puzzles and the game is commendably logical so reasonably easy (what I have seen so far, anyway). I was very impressed with Kayleth; it offers a good game for the occasional adventurer due to its graphics and logicality, while the hardened player will appreciate its sophisticated features and enjoyable plot. Very narrowly misses Monster Hit status.

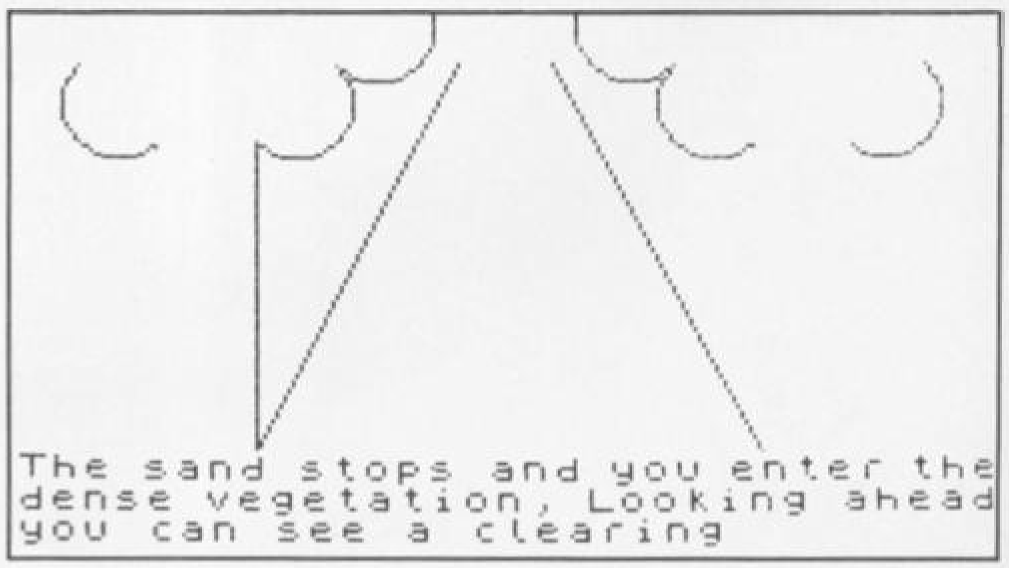
GREAT

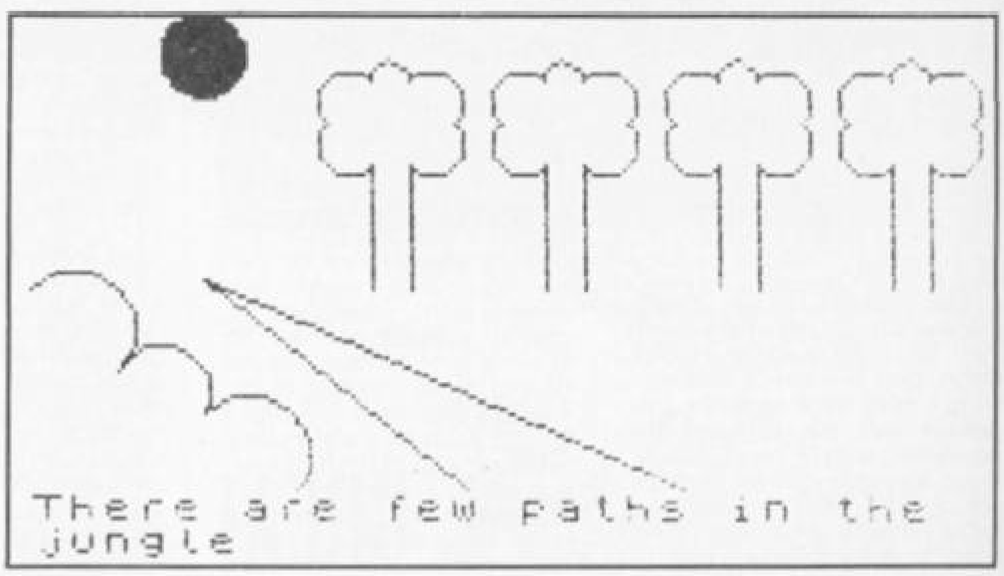
KINGDOM OF KULL



BY JEREMY WILLIAMS

Travelling back in time in search of the lost crystal, you finally track it down to a castle in the mysterious Kingdom of Kull . . .





Listing 1

This is the Basic loader program, which also contains instructions for the game. Type this in and SAVE it with the command: SAVE "Kingdom" LINE 1.

Listing 2

The main game listing. Once you've typed it all in, you should SAVE it onto tape immediately after listing one, using the command: SAVE "King" LINE 1. Then when you want to play the game you just rewind the tape and load using LOAD "" and the game will autorun.

Listing 1

```
  0>LET v=PEEK 23613:POKE 23613
,v-2:POKE 23609,50:POKE 23658,8:
PAPER 6:BORDER 2:CLS :PRINT AT 1
5,10;FLASH 1;BRIGHT 1;INK 7;PAPE
R 3;"STOP TAPE":LET a$="
Kingdom of Kull
"
  10 FOR n=24 TO 1 STEP -1: BEEP
 .1,10
  20 INK 1: PRINT AT 5,0;a$(n TO
 ): NEXT n
  30 PRINT AT 10,7; Jeremy Wil
liams"
  40 PRINT #1;"Do you want instr
uctions ?"
  50 IF INKEY$="" THEN  GO TO 50
  60INKEY$="N" THEN  GO TO 1000
  70 BEEP .5,0: CLS : PRINT AT 2
,2;"Nafir, an escaping prisoner
  from the 28th century was
  passing through time when he
  dropped a beacon, in the form
ofa crystal, sending out vital
  information to enemy galaxies.
.. He was caught while re-enteri
ngreal time and was questioned.H
e gave the time-zone and also th
e place.........The castle in th
e        ""Kingdom of Kull"""
  80 PRINT #1;"Press any key to
continue ?": PAUSE 0: CLS
  90 BEEP .5,0: CLS : PRINT AT 2
,2;"You, Morner a 28th century
  historian have been sent back
intime to destroy the crystal.Yo
u must seek the castle of Kull
  where the king keeps the cryst
alcollecting items and using the
m to help you on your journey.
   When you have found the cryst
alyou must return to your own ti
meusing one of the objects.You
  will not have long before the
  crystal explodes so be ready.
  And beware, some items have mo
rethan one use..."
 100 PRINT #0;"Do you wish to kn
ow the words   that the program
takes ?": PAUSE 0: IF INKEY$="N"
 THEN  GO TO 150
 110 BEEP .5,0: CLS : PRINT AT 1
,10;"WORD LIST";AT 1,10; OVER 1;
"_________": PRINT AT 3,0;"OPEN
  (object)                 CLOSE
  (object)";AT 6,0;"TAKE     (ob
ject)                DROP    (ob
ject)";AT 9,0;"GIVE     (person)
   (object)   ASK      (person)
FOR (object)";AT 12,0;"EXAMINE (
object)";AT 14,0;"SLEEP -refresh
s you,very useful";AT 16,0;"KILL
   (person) -Don't get too
              carried away";AT 1
9,0;"HOW -strength and inventory
";AT 21,0;"WAIT"
 150 PAUSE 0: BEEP .5,0: CLS : P
RINT AT 1,7;"Some special things
              you should know...
"
 160 PRINT AT 5,5;"QUIT -gives c
hoice of                  saving
present game";AT 8,5;"LOAD -load
s a saved game              back
 into computer";AT 11,5;"!    -r
epeats last command"
 165 PRINT AT 14,5;"Some entranc
es may be      locked and need t
o be opened     with something.
                eg:-  ""OPEN DOOR
 WITH KEY"""
 166 PRINT AT 19,1;"Graphics can
 be turned ON / OFF"
 170 PRINT #1;"Press any key to
continue": PAUSE 0: IF INKEY$="^
" THEN  SAVE "KINGDOM" LINE 0
 1000 CLS : PRINT AT 5,0;"When th
e program has loaded, andyou are
 shown the picture, pressa key t
o be given the exits and informa
tion.": LET v=PEEK 23613: POKE 2
3613,v+2: BEEP 1,0: PRINT AT 20,
10;"Start tape"
 8400 LOAD "KING"
```

Listing 2

```
   1 REM J.D.WILLIAMS 1984
  10 LET s=10: LET xy=0: LET K=0
: LET H$="": BORDER 0: PAPER 7:
INK 0: CLS : PRINT AT 10,10;"Ple
ase wait": LET u=0: LET p=0: LET
 st=20: DIM e(10,3): DIM d(20,4)
: DIM e$(10,9): DIM d$(20,9)
  20 PRINT AT 1,8;"KINGDOM OF KU
LL": LET X=2: LET Y=7
  30 RESTORE 9830: FOR F=1 TO 10
: READ G$: LET E$(F)=G$: NEXT F
  40 FOR F=1 TO 20: READ G$: LET
 D$(F)=G$: NEXT F
 150 RESTORE 9800: FOR N=1 TO 10
: FOR M=1 TO 3: READ F
 160 LET e(N,M)=F
 170 NEXT M: NEXT N
 180 FOR N=1 TO 20: FOR M=1 TO 4
: READ F
 190 LET e(N,M)=F
 200 NEXT M: NEXT N
 210 GO SUB 9999: PRINT AT 21,0;
: POKE 23692,255: LET g$="LOOK "
: LET K=0: GO TO 442
 400 LET N=x: LET M=y
 410 LET Y=Y+(g$(1)="S" AND ss=1
)+(2 AND g$(1)="D" AND dd=1)-(g$
(1)="N" AND nn=1)-(2 AND g$(1)="
U" AND uu=1): LET x=x+(g$(1)="E"
 AND ee=1)-(g$(1)="W" AND ww=1)
 440 IF N=x AND M=y THEN  PRINT
"The way ";g$;" is blocked": GO
TO 800
 442 LET A=(100*x)+(10*y): CLS
 443 GO SUB 9999: GO SUB 7000+A+
U: INK 0: PAPER 7: PRINT AT 15,0
;G$
 445 PAUSE 0: GO SUB 9890: PRINT
 "You see-"
 450 GO SUB 9900
 460 IF a=890 THEN  GO TO 4000
 472 IF a=620 OR a=920 THEN  GO
TO 3000
 480 GO TO 500
 500 POKE 23692,253: PRINT : LET
 ST=ST-.2: LET sk=(9-p)+3*(d(1,4
)=1): IF st<0 THEN  GO TO 9500
 510 IF R>2 AND RND>.8 THEN  PRI
NT "The ";e$(R);" attacks you":
LET I=R: GO TO 1240
 520 LET o$="": LET s$="": INPUT
 "-"; LINE g$
 530 IF g$="" OR g$="WAIT" THEN
 PRINT "You are waiting.": GO TO
 800
 540 IF g$="!" THEN  LET g$=h$
 550 LET H$=G$: LET g$=g$+" "
 555 IF g$="LOOK " THEN  GO TO 4
42
 556 IF g$="ON " THEN  LET u=0:
GO TO 500
 557 IF g$="OFF " THEN  LET u=5:
 GO TO 500
 558 IF g$="SLEEP " THEN  GO TO
1100
 560 IF g$(1)="N" OR g$(1)="S" O
R G$(1)="W" OR G$(1)="E" AND G$(
2)<>"X" AND G$(2)<>"A" OR G$(1)=
"U" OR G$(1)="D" AND G$(2)<>"R"
THEN  GO TO 400
 562 IF g$="PAUSE " THEN  BORDER
 2: PAUSE 0: BEEP .5,20: BORDER
7: GO TO 500
 564 IF g$="QUIT " THEN  GO TO 2
200
 565 IF g$="LOAD " THEN  GO TO 8
500
 566 IF g$="HOW " THEN  GO TO 17
00
 570 FOR n=1 TO 8: IF g$(n)=" "
THEN  GO TO 580
 575 NEXT N: PRINT "WHAT ??": GO
 TO 800
 580 LET v$=g$(1 TO n-1)
 590 IF LEN g$<=n THEN  PRINT v$
;" what?": GO TO 500
 600 FOR m=n+1 TO n+9: IF G$(M)=
" " THEN  GO TO 610
 605 NEXT M: PRINT "WHAT ??": GO
 TO 500
 610 LET s$=g$(n+1 TO m-1)
 615 IF LEN s$<9 THEN  LET s$=s$
+" ": GO TO 615
 617 IF v$(2)="X" AND s$(4)="G"
THEN  LET xy=1
 620 RESTORE 9850: FOR n=2 TO 13
: READ W$: IF v$=W$ THEN  GO TO
1000+(100*n)
 630 NEXT n
 640 PRINT "I do not know the ve
rb ";v$: GO TO 800
 810 LET N=INT (RND*7)+3: IF E$(
N)="DEAD-BODY" THEN  GO TO 500
 820 LET m=INT (RND*3)-1: LET f=
INT (RND*3)-1
 830 LET e(N,1)=e(N,1)+m AND e(n
,1)<>0 AND e(n,2)<>11: LET e(N,2
)=e(N,2)+f AND e(n,2)<>0 AND e(n
,2)<>11
 850 IF e(N,1)-m=x AND e(N,2)-F=
y AND m<>0 OR E(N,1)-m=X AND E(N
,2)-f=Y AND f<>0 THEN  PRINT "Th
e ";e$(N);" leaves": LET R=0
 860 IF e(N,1)=x AND e(N,2)=y AN
D m<>0 OR E(N,1)=X AND E(N,2)=Y
AND f<>0 THEN  PRINT "The ";e$(N
);" enters": LET R=N
 870 GO TO 500
1110 IF a=800 AND d(9,4)=1 THEN
 GO TO 2600
1120 PRINT "You have a rough nig
ht but sleep soundly": LET st=st
+3
1140 LET s=s-1: IF s<=0 THEN  GO
 TO 1150
1145 PRINT "You wake late in the
 morning": GO TO 800
1150 LET I=INT (RND*7)+3: IF e$(
I)="DEAD-BODY" THEN  GO TO 1145
1160 LET e(I,1)=x: LET e(I,2)=y:
 PRINT "You are awakened by a th
umping  and the ";e$(I);" enters
"'"He sees you and attacks!": GO
 TO 1240
1200 FOR I=1 TO 10: IF e(I,1)=x
AND e(I,2)=y THEN  GO TO 1220
1210 NEXT I: LET I=0: PRINT "The
re is nobody to kill": GO TO 800
1220 IF s$<>e$(I) THEN  PRINT s$
;"is not here": GO TO 800
1240 LET V=10: LET z=I+5: IF I=1
 OR I=2 THEN  LET Z=15
1260 LET m=RND*5: LET f=RND*5: L
ET sk=sk+m: LET z=z+f
1265 IF sk>z THEN  LET V=V-2: PR
INT "You slash him violently"
1270 IF sk<z+4 THEN  GO TO 1285
1275 PRINT "With one swipe you c
leave his   head"
1277 LET e$(I)="DEAD-BODY": IF e
(I,3)<>0 THEN  GO TO 2500
1280 IF I=R THEN  LET R=0: LET I
=0
1281 GO TO 800
1285 IF sk<z AND sk>=z-3 THEN  L
ET st=st-1: PRINT "He stabs you"
1290 IF sk<z-3 THEN  LET st=st-2
: PRINT "His sword gashes your c
hest"
1295 IF st<0 THEN  PRINT "You ar
e killed by the ";E$(I): GO TO 9
500
1310 GO SUB 4200
1330 LET z=z-f: LET sk=sk-m: IF
V<0 THEN  PRINT "He slips to the
 ground, dead": GO TO 1277
1340 INPUT INKEY$="Y": GO TO 126
0
1400 IF LEN g$<M+5 THEN  PRINT "
Ask ";s$;" for what?": GO TO 800
1405 LET o$=g$(m+5 TO )
1410 IF LEN o$<9 THEN  LET o$=o$
+" ": GO TO 1410
1415 FOR n=1 TO 10: IF s$=e$(n)
AND e(n,1)=x AND e(n,2)=y THEN
GO TO 1430
1420 NEXT n: PRINT s$;" is not h
ere": GO TO 800
1430 FOR m=1 TO 10: IF o$=d$(m)
AND e(n,3)=m THEN  GO TO 1450
1440 NEXT M: PRINT "The ";s$;" d
oes not have a   ";o$: GO TO 800
1450 IF RND>.5 THEN  GO TO 1470
1460 PRINT "The ";s$;" will not
give it  to you": GO TO 800
1470 PRINT "The ";s$;" gives you
 the     ";o$: LET e(n,3)=0
1480 LET K=K+1: LET d(m,4)=1: GO
 TO 800
1500 IF a=870 OR a=990 OR a=1000
 THEN  GO SUB 2700
1510 FOR n=11 TO 20: IF s$<>d$(n
```

```
) THEN  NEXT n: PRINT "You can't
 open the ";s$: GO TO 800
1520 FOR n=11 TO 20: IF d(n,1)=x
 AND d(n,2)=y THEN  GO TO 1540
1530 NEXT n: PRINT "There isn't
a ";s$;" here": GO TO 800
1540 IF d(n,4)=1 THEN  PRINT "Th
e ";d$(n);" is already open": GO
 TO 800
1550 PRINT "You open the ";s$;".
Inside is ";: IF d(n,3)=0 THEN
PRINT "nothing": GO TO 1570
1560 LET m=d(n,3): PRINT "the ";
d$(m): LET D(N,4)=1: LET D(m,3)=
0: LET d(n,3)=0
1570 IF a=990 OR a=230 OR a=1000
 OR a=870 THEN  PRINT "You can n
ow go ";
1572 IF A=230 THEN  LET NN=1: PR
INT "North"
1573A=1000 OR a=870 THEN  LET ee
=1: PRINT "East"
1574 IF a=990 THEN  LET ww=1: PR
INT "West"
1580 GO TO 800
1600 FOR n=11 TO 20: IF s$<>d$(n
) THEN  NEXT n: PRINT "You can't
 close a ";s$: GO TO 800
1610 FOR n=11 TO 20: IF d(n,1)=x
 AND d(n,2)=y THEN  GO TO 1620
1615 NEXT n: PRINT "There isn't
a ";s$;" here": GO TO 800
1620 IF d(n,4)=0 THEN  PRINT "Th
e ";d$(n);" is already closed":
GO TO 800
1630 PRINT "You close the ";s$:
LET d(n,4)=0: GO TO 800
1700 PRINT "You are feeling ";("
very strong" AND st>=20);("stron
g" AND st>=15 AND st<20);("healt
hy" AND st>=10 AND st<15);("weak
" AND st>=5 AND st<10);("very we
ak" AND st<5)
1710 PRINT '"You have:-"'
1720 FOR n=1 TO 10: IF d(n,4)=1
THEN  PRINT TAB 11;"The ";d$(n)
1730 NEXT n: GO SUB 4200: GO TO
800
1800 FOR N=1 TO 10: IF S$=D$(N)
AND D(N,1)=X AND D(N,2)=Y THEN
GO TO 1820
1810 NEXT n: PRINT "The ";s$;" i
s not here": GO TO 800
1820 IF K=5 THEN  PRINT "You can
't carry the ";s$: GO TO 800
1830 PRINT "You take the ";s$: L
ET d(n,1)=0: LET d(n,2)=0: LET d
(n,4)=1: LET K=K+1: GO TO 800
1900 FOR n=1 TO 10: IF s$=d$(n)
AND d(n,4)=1 THEN  GO TO 1920
1910 NEXT n: PRINT "You haven't
got a ";s$;"to   drop": GO TO 80
0
1920 PRINT "You drop the ";s$: L
ET d(n,1)=x: LET d(n,2)=y: LET d
(n,4)=0: LET k=k-1: GO TO 800
2000 IF LEN g$<=m+2 THEN  PRINT
"Give ";s$;" what?": GO TO 800
2005 LET o$=g$(m+1 TO )
2010 IF LEN o$<9 THEN  LET o$=o$
+" ": GO TO 2010
2015 FOR n=1 TO 10: IF o$=d$(n)
AND d(n,4)=1 THEN  GO TO 2030
2020 NEXT n: PRINT "You don't ha
ve a ";o$;" to   give": GO TO 80
0
2030 FOR m=1 TO 10: IF s$=e$(m)
AND e(m,1)=x AND e(m,2)=y THEN
GO TO 2050
2040 NEXT m: PRINT s$;" is not h
ere": GO TO 800
2050 IF a=1090 AND o$(4)="G" AND
 s$(1)="K" THEN  GO TO 2400
2060 PRINT "You give the ";s$;"
the ";o$: PRINT "The ";s$;" says
 ""Thank you""": IF e(m,3)<>0 OR
RND<.25 THEN  PRINT "The ";s$;"
doesn't want it"'"and gives it
back": GO TO 800
2070 LET K=K-1: LET e(m,3)=n: LE
T d(n,4)=0
2080 IF a=120 AND o$(3)="S" AND
s$(1)="B" THEN  GO TO 2450
2090 GO TO 800
2100 LET k=k-1: IF S$(2)="I" THE
N  LET d(8,4)=0: PRINT "The fish
 is unbearable but you  eat it":
 LET st=st+4: GO TO 800
2110 IF s$(2)="O" AND d(2,4)=1 T
HEN  LET d(2,4)=0: PRINT "You ea
t quickly": LET st=st+7: GO TO 8
00
2120 IF s$(2)="O" AND D(7,4)=1 T
HEN  LET d(7,4)=0: PRINT "You ea
t heartily": LET st=st+9: GO TO
800
2130 LET k=k+1: IF s$(2)<>"O" AN
D s$(2)<>"I" THEN  PRINT "You ca
n't eat ";s$: GO TO 800
2140 PRINT "You haven't got any
";s$: GO TO 800
2200 PRINT "Save game (y/n) ?":
PAUSE 0: IF INKEY$="N" THEN  GO
TO 2250
2220 DIM F(3): LET F(1)=X: LET F
(2)=Y: LET F(3)=ST: SAVE "KINGDA
TA" DATA D(): SAVE "-" DATA E():
 SAVE "-" DATA F(): SAVE "-" DAT
A D$(): SAVE "-" DATA E$(): GO T
O 500
2250 IF INKEY$="" THEN  GO TO 40
50
2260 GO TO 2250
2300 FOR n=1 TO 20: IF s$=d$(n)
AND d(n,1)=x AND d(n,2)=y THEN
RESTORE 2350+n: READ g$: PRINT "
The ";s$;" is ";g$: GO TO 800
2310 NEXT n: PRINT "The ";s$;"is
n't here": GO TO 800
2351 DATA "sharp"
2352 DATA "delicious"
2353 DATA "strong"
2354 DATA "small"
2355 DATA "fit for a king,and yo
u notice a small button   set on
 the side"
2356 DATA "dry"
2358 DATA "smelly"
2359 DATA "sturdy,  and   comfor
table to lie in"
2360 DATA "useful"
2375 DATA "nothing special"
2400 LET WW=1: PRINT "The King o
f Kull accepts your   gift and i
n return gives you    some food"
: LET d(7,4)=1: LET D(5,4)=0: PR
INT "He orders his guards to let
 you pass west": LET E(1,3)=5: G
O TO 800
2450 PRINT "The Bear is delighte
d with your gift and as he moves
 you notice an opening leading n
orth": LET nn=1: GO TO 800
2500 PRINT "You search the dead-
body and    find a ";d$(e(i,3))
2510 LET d(e(i,3),4)=1: LET e(i,
3)=0: GO TO 800
2600 LET K=K-1: PRINT "You fall
asleep in the boat, Youawake som
ewhere high up in the  mountains
 !": PAUSE 0: LET d(9,4)=0: LET
x=5: LET y=8: GO TO 442
2700 IF LEN g$>=M+8 THEN  LET o$
=g$(m+6 TO ): GO TO 2720
2705 IF a=870 THEN  PRINT "The c
urtain burns your hands": LET st
=st-3
2710 PRINT "You cannot open this
 entrance": GO TO 800
2720 IF LEN O$<9 THEN  LET O$=O$
+" ": GO TO 2720
2730 FOR n=1 TO 10: IF o$=d$(n)
THEN  GO TO 2750
2740 NEXT n: PRINT "You can't op
en a door with a    ";o$: GO TO
800
2750 IF d(n,4)=0 THEN  PRINT "Yo
u are not carrying a ";o$: GO TO
 800
2760 IF a=870 AND o$(3)="P" THEN
  RETURN
2770 IF a=990 AND o$(1)="K" THEN
  RETURN
2780 IF a=1000 AND o$( TO 3)="BO
N" THEN  RETURN
2790 GO TO 2710
3000 PRINT "It is dark in the fo
rest": INPUT INKEY$="Y": IF d(10
,4)=1 THEN  GO TO 3020
3010 PRINT "You stumble around h
elplessly   and an unknown enemy
 comes up    and chops your head
 off !!": GO TO 9500
3020 PRINT "But you light your l
antern and  the path is clear":
GO TO 800
3100 PAPER 7: INK 0: PRINT AT 15
,0;Q$: PAUSE 0: LET Q$="You slip
 and bang your head,    Strangel
y, when you awake you   are in d
ifferent surroundings": PRINT AT
 15,0;Q$: IF a=110 THEN  LET x=5
: LET y=2
3110 IF a=470 THEN  LET x=5: LET
 y=10
3130 PAUSE 0: GO TO 442
4000 PAUSE 0: PAPER 0: CLS : IF
D(5,4)=1 AND XY=1 THEN  PRINT "Y
ou rember the button in'your   r
ing and press it.Instantly you a
re back in your own time where y
ou arrive to a hero's welcome":
GO TO 4030
4010 PRINT "You don't know how t
o return to your own time and wh
ile you are thinking the crystal
 explodes": GO TO 9500
4030 FOR f=10 TO 30 STEP 2: BEEP
 .05,F: BEEP .02,0: NEXT F
4040 PAUSE 0: CLS : PAPER 5: GO
SUB 9990: LET i=6: GO SUB 9950:
PAPER 1: INK 2: FOR f=7 TO 14: P
RINT AT f,3;"     ": NEXT f: PLO
T 24,56: DRAW 0,64: DRAW 40,0: D
RAW 0,-64: DRAW -40,0: FOR f=1 T
O 6: PLOT 38+f,120: DRAW 6,0,-PI
: NEXT f
4041 FOR f=29 TO 60 STEP 20: PLO
T f,100: DRAW 10,0: DRAW 0,10: D
RAW -10,0: DRAW 0,-10: NEXT f
4042 FOR g=120 TO 200 STEP 80: F
OR f=0 TO 18 STEP 3: PLOT g+f,70
: DRAW 0,70: NEXT f: NEXT g
4043 DRAW 10,10: DRAW -120,0: DR
AW 10,-10: DRAW 100,0: PLOT 120,
70: DRAW 18,0: PLOT 200,70: DRAW
 18,0
4050 LET m=-20: LET n=20: PRINT
#1;"Do you wish to try again ? "
4060 IF m=-20 THEN  FOR f=m TO n
+1
4065 IF m=20 THEN  FOR f=m TO n+
1 STEP -1
4070 IF INKEY$="Y" THEN  GO TO 1
4080 IF INKEY$="N" THEN  RANDOMI
ZE USR 0
4090 BEEP .2,f: NEXT f
4095 LET n=-n: LET m=-m
4100 GO TO 4060
4200 LET P=(-2 AND ST>20)+(1 AND
 ST<15)+(2 AND ST<10)+(2 AND ST<
5): RETURN
7110 BORDER 0: PAPER 0: GO SUB 9
990
7115 LET Q$="The hole is tuo dar
k to see.": GO TO 3100
7120 GO SUB 7427
7125 LET Q$="You are in a bare r
oom": LET SS=1: RETURN
7130 BORDER 2: PAPER 2: GO SUB 9
990: PLOT 40,130: DRAW 10,-60,2/
-PI: DRAW 40,0,2/-PI: DRAW -5,20
: DRAW -45,40,-PI/2
7135 LET NN=1: LET Q$="You are i
n a well furnished room": RETURN
7140 LET i=6: PAPER 5: GO SUB 99
90: GO SUB 9950: INK 0: FOR f=1
TO 8: PLOT 0,100+f: DRAW 70,-5:
DRAW 60,0: DRAW 60,4: DRAW 65,-3
: NEXT f
7141 PLOT 75,95: DRAW -20,-30: P
LOT 70,85: DRAW 5,-10: PLOT 200,
97: DRAW -10,-20: DRAW 5,-10: PL
OT 193,70: DRAW -5,-5
7145 PAPER 7: INK 0: PRINT AT 15
,0;"You cross the mountains to t
he other side. A  steep ravine
madeslippery by the rain bars yo
ur  path."
7146  PRINT "You attempt to cros
s it...."
7147 PAUSE 0: PAPER 2: CLS : PAU
SE 8: PAPER 7: CLS : PRINT AT 15
,0;"You slip on a wet rock&and
    plummet down to the rocks
    beneath": BEEP 1,-40: GO TO
9500
7150 PAPER 0: INK 7: BORDER 0: G
```

```
O SUB 9990: PLOT 100,120: DRAW 2
5,-5,PI: DRAW 30,2,PI: DRAW 25,0
,PI: DRAW 30,15,PI: DRAW -20,20,
PI: DRAW -30,-3,PI: DRAW -28,4,P
I
7151 DRAW -20,-5,PI: DRAW -25,0,
PI: DRAW 12,-28,PI
7152 PLOT 127,108: DRAW -15,-15:
 DRAW 3,0: DRAW -15,-15: DRAW 1,
0: DRAW 16,16: DRAW -3,0: DRAW 1
5,15
7155 LET SS=1: LET NN=1: LET Q$=
"Up above the valley a gigntic
 storm comes into view , You are
 soon drenched to the skin": LET
 st=st-1: RETURN
7160 BORDER 1: PAPER 7: INK 0: G
O SUB 9990: LET i=2: GO SUB 9950
7161 INK 0: PLOT 128,100: DRAW 6
4,35,.5/-PI: DRAW 63,25,.9/PI: P
LOT 128,100: DRAW -128,75,.3*-PI
7162 PLOT 128,100: DRAW -30,-44,
.3*PI: PLOT 128,100: DRAW 15,-44
,.4*PI
7165 LET NN=1: LET SS=1: LET EE=
1: LET Q$="You are on a narrow w
inding paththat leads down to a
dim valley": RETURN
7170 BORDER 4: PAPER 5: GO SUB 9
990: LET i=6: GO SUB 9950: PLOT
127,56: DRAW 0,64: DRAW -127,0:
FOR n=7 TO 14: PRINT AT n,0; PAP
ER 0;"                  ": NEXT n
7171 PRINT AT 9,3; PAPER 2;"  ";
AT 10,3;"  ";AT 9,11;"  ";AT 10,
11;"  ": INK 0: PLOT 32,88: DRAW
 0,15: DRAW -1,0: DRAW 0,-15: PL
OT 96,88: DRAW 0,15: DRAW -1,0:
DRAW 0,-15
7172 PLOT 24,96: DRAW 80,0: DRAW
 0,-1: DRAW -80,0 PRINT AT 14,16
; PAPER 4;"                ";AT
13,16;"                ": INK 4:
 FOR f=1 TO 8: PLOT 128,65+f: DR
AW 127,25,2/PI: NEXT f
7173 PRINT AT 12,26; PAPER 4;"
    ";AT 11,29;"   ";AT 10,31;"
": INK 0
7175 LET ee=1: LET NN=1: LET Q$=
"You are outside a derelict hut
 in the middle of nowhere": RETU
RN
7180 GO SUB 7186
7185 INK 0: LET ss=1: LET dd=1:
LET Q$="You are at the top of th
e tree. the climb was hard but t
he view is lovely": LET st=st-1:
 RETURN
7186 BORDER 5: PAPER 5: GO SUB 9
990: LET i=6: GO SUB 9950:
7187 INK 4: FOR f=1 TO 8: PLOT 0
,86-f: DRAW 100,-10,3/-PI: DRAW
75,5,2/-PI: DRAW 80,5,-5/PI: NEX
T f: PRINT AT 13,0; PAPER 4;"
                                ";A
T 12,0;"              ";AT 12,14;
"                   ";AT 11,1;"
      ";AT 11,23;"          ";AT
10,25;"      "
7188 PRINT AT 11,5; INK 0; PAPER
 4;"     ";AT 13,25;"
"
7189 RETURN
7190 GO SUB 7186
7193 PAPER 7: PAUSE 0: FOR f=1 T
O 30: PRINT : INPUT INKEY$="y":
NEXT f: PRINT
7195 INK 0: PAPER 7: PRINT AT 15
,0;"If you have come up a tree t
he  only way back is down. You s
tep out into empty space, fall a
nd  crack your skull": GO TO 950
0
7200 GO SUB 7206
7205 LET uu=1: LET EE=1: LET Q$=
"You are underneath a massive
 tree. It has branches everywher
eand looks reasonably easy to
 climb.": RETURN
7206 BORDER 4: PAPER 4: GO SUB 9
990: LET i=2: GO SUB 9950:
7207 INK 0: PLOT 100,56: DRAW 0,
119: PLOT 155,56: DRAW 0,119: PL
OT 120,125: DRAW -40,30,-PI/2: D
RAW 40,-20,PI/2: PLOT 100,129: O
VER 1: DRAW 0,5: PLOT 155,76: DR
AW 0.8: OVER 0
7208 PLOT 120,90: DRAW 50,20,PI:
 DRAW -50,-5,-PI
7209 RETURN
7211 RESTORE 7210: FOR f=1 TO 5:
 READ n,m: PLOT n,m: DRAW -20,40
,2/PI: PLOT n,m: DRAW 10,30,2/-P
I: PLOT n,m: DRAW -30,40,2/PI: P
LOT n,m: DRAW 15,20,2/-PI: NEXT
f
7215 PRINT AT 15,0;"You advance
carefully across themarsh but be
fore you are halfwayyou start to
 sink down slowly...You try to s
top yourself but    there is not
hing you can do": GO TO 9500
7219 DATA 50,70,150,80,200,100,1
50,120,80,110
7220 GO SUB 8098
7225 LET Q$="The door locks and
you find       yourself in a room
with a table": LET ww=1: RETURN
7230 GO SUB 8068
7235 LET q$="The path stops by a
 house": LET ee=1: RETURN
7245 GO SUB 7246: LET ee=1: RETU
RN
7246 BORDER 0: PAPER 0: GO SUB 9
990: LET q$="It is dark and you
must feel     your way around": R
ETURN
7255 LET nn=1: LET ee=1: GO SUB
7246: RETURN
7260 PLOT 128,90: DRAW -100,50,.
4*-PI: DRAW -28,35,2/PI: PLOT 12
8,90: DRAW 40,50,.4*PI: DRAW 80,
35,.4*PI
7261 GO SUB 7266
7265 LET ee=1: LET ww=1: LET G$=
"You are walking along the botto
mof a dim valley filled with
 trees": RETURN
7266 PLOT 128,90: DRAW -40,-34,2
/PI: PLOT 128,90: DRAW 40,-34,2/
-PI: RETURN
7270 BORDER 6: PAPER 6: GO SUB 9
990: PLOT 0,116: DRAW 240,-59,2/
PI: PLOT 255,130: DRAW -165,-55,
3/PI: LET i=3: GO SUB 9950
7275 LET ss=1: LET ww=1: LET Q$=
"You are standing in a lonely
 wasteland. Far,far,far away to
 the East a great forest stands"
: RETURN
7280 BORDER 6: PAPER 6: GO SUB 9
990: LET I=3: GO SUB 9950:
7281 INK 0: PLOT 0,56: DRAW 128,
0,-PI/2: DRAW 127,0,-PI/3: PLOT
60,83: DRAW 195,30,-PI/4: PLOT 8
8,100: DRAW -88,10,PI/2
7285 LET NN=1: LET SS=1: LET Q$=
"You are in a desert. The sand
 stretches on as far as you can
 see in all directions": RETURN
7290 BORDER 6: PAPER 6: GO SUB 9
990: LET I=2: GO SUB 9950
7291 PLOT 0,70: DRAW 120,-4,-(2/
PI): PLOT 90,56: DRAW 160,100,2/
PI: PLOT 200,110: DRAW -150,-31,
3/PI
7295 LET nn=1: LET ee=1: LET Q$=
"You trudge on wearily, the sand
 still goes on and on": RETURN
7300 GO SUB 7688
7301 INK 0: PLOT 0,90: DRAW 100,
10: DRAW 50,-5: DRAW 105,10:
7305 LET ee=1: LET ww=1: LET Q$=
"A grassy moor stretches out in
 front of you": RETURN
7310 GO SUB 7391
7315 LET q$="Ahead the trees sto
p and there  is a clearing": LET
 WW=1: LET ss=1: RETURN
7320 GO SUB 7406
7325 LET ss=1: LET nn=1: LET q$=
"You are in a patch of trees": R
ETURN
7330 BORDER 4: PAPER 4: GO SUB 9
990: GO SUB 7266
7335 LET q$="You are on a rough
track": LET WW=1: LET nn=1: RETU
RN
7345 LET ee=1: LET ss=1: GO SUB
7246: RETURN
7355 LET ww=1: GO SUB 7246: RETU
RN
7360 BORDER 0: PAPER 7: GO SUB 9
990: PLOT 0.56: DRAW 40.20: PLOT
 255,56: DRAW -40,20
7361 FOR f=1 TO 230 STEP 40: PLO
T f,70+(RND*20): DRAW 30,0,-PI:
NEXT f: FOR g=10 TO 20 STEP 10:
FOR f=g TO 230-(30-g) STEP 40: P
LOT f,70+(g*3)+(RND*20): DRAW 30
,0,-PI: NEXT f: NEXT g
7362 FOR f=1 TO 10: CIRCLE 148,1
00,f: CIRCLE 147,100,f: NEXT f
7365 LET ss=1: LET ww=1: LET Q$=
"Your path is blocked by a pile
 of rocks , You search for a way
 through and find a tunnel
 leading south": RETURN
7370 BORDER 4: PAPER 4: GO SUB 9
990: INK 0: CIRCLE 128,116,59
7371 PLOT 175,150: DRAW -25,-90,
PI/1.5
7372 PLOT 140,120: DRAW -65,20,2
/PI: PLOT 150,135: DRAW -60,25,2
/PI: PLOT 135,95: DRAW -65,25,2/
PI
7375 LET nn=1: LET ss=1: LET ee=
1: LET Q$="You continue on down
the tunnel,To the south there is
 a shallow alcove": RETURN
7380 BORDER 4: PAPER 4: INK 0: G
O SUB 9990: PLOT 80,56: DRAW 0,6
8: DRAW 100,0,-PI: DRAW 0,-68
7381 IF d(3,1)=3 AND d(3,2)=8 AN
D d(3,4)=0 THEN  FOR f=1 TO 6 ST
EP 5: PLOT 128,80+f: DRAW 25,-20
,PI: DRAW -10,30,PI: DRAW 0,-25,
PI: DRAW 5,15,PI: DRAW 0,-10,PI:
 DRAW 0,5,PI: NEXT f
7385 LET ee=1: PAPER 7: INK 0: P
RINT AT 15,0;"There is not much
to see apart  from the solid roc
k wall";: IF d(3,1)=3 AND d(3,2)
=8 THEN  PRINT " and a  rope whi
ch is lying on the      floor"
7386 LET Q$="": GO TO 445
7390 GO SUB 7391: GO TO 7395
7391 PAPER 4: GO SUB 9990: PLOT
48,56: DRAW 0,79: DRAW 20,20,PI:
 DRAW 20,20,PI: PLOT 208,56: DRA
W 0,79: DRAW -20,20,-PI: DRAW -2
0,20,-PI: PLOT 230,135: DRAW 20,
20,PI: PLOT 24,135: DRAW -20,20,
-PI
7392 FOR N=5 TO 14: PRINT AT N,3
; PAPER 0;"   ";AT N,26;"   ": N
EXT N: PLOT 48,56: DRAW 50,100:
PLOT 208,56: DRAW -50,100: RETUR
N
7395 LET ww=1: LET ss=1: LET Q$=
"The sand stops and you enter th
edense vegetation, Looking ahead
 you can see a clearing": RETURN
7400 GO SUB 7406
7401 PLOT 0,104: DRAW 20,-20,-PI
: DRAW 20,-10,-PI: DRAW 20,-20,-
PI:
7404 PLOT 100,56: DRAW -60,50: P
LOT 150,56: DRAW -110,50
7405 LET NN=1: LET WW=1: LET Q$=
"There are few paths in the
 jungle": RETURN
7406 BORDER 4: PAPER 4: GO SUB 9
990: LET i=6: GO SUB 9950: INK 0
: FOR f=80 TO 200 STEP 40: PLOT
f-1,104: DRAW 0,30: DRAW -10,10,
-PI: DRAW 10,10,-PI: DRAW 9,0,-P
I: DRAW 10,-10,-PI: DRAW -10,-10
,-PI: DRAW 0,-30: NEXT f
7407 FOR f=10 TO 25 STEP 5: FOR
n=1 TO 4: PRINT AT 4+n,f; PAPER
0;" ": NEXT n: NEXT f: RETURN
7410 PAPER 6: GO SUB 9990: LET i
=2: GO SUB 9950:: PLOT 25,145: D
RAW 30,0: DRAW 0,30: DRAW -30,0:
 DRAW 0,-30:
7411 INK 0: PLOT 0,100: DRAW 255
,0: DRAW 0,-2: DRAW -255,0: DRAW
 0,2: PLOT 150,56: DRAW 0,75: PL
OT 165,56: DRAW 0,75: FOR f=10 T
O 70 STEP 10: PLOT 150,56+f: DRA
W 15,0: NEXT f
7412 PLOT 5,110: DRAW 10,0: PLOT
 25,106: DRAW 20,5: PLOT 50,120:
 DRAW -10,-5: PLOT 75,110: DRAW
10,-2: PLOT 90,115: DRAW 6,-1: P
LOT 125,108: DRAW 13,4: PLOT 130
,100: DRAW 8,4: PLOT 170,114: DR
AW 10,-4: PLOT 190,120: DRAW 16,
-4: PLOT 230,130: DRAW 17,-5: PL
OT 240,114: DRAW -13,-3
```

```
7415 LET dd=1: LET Q$="You climb
 the ladder and find   a comfort
able bed in the hay.Youcan see t
he evening sun through a window"
: RETURN
7420 GO SUB 7427
7425 LET ss=1:ee=1: LET Q$="You
are inside the barn.There isnoth
ing in it but hay": RETURN
7427 BORDER 6: PAPER 6: GO SUB 9
990: PLOT 40,56: DRAW 0,74: DRAW
 -40,45: PLOT 215,56: DRAW 0,74:
 DRAW 40,45: DRAW -40,-45: DRAW
-175,0: FOR F=40 TO 200 STEP 19:
 PLOT F,130: DRAW 0,-74: NEXT F:
 RETURN
7430 BORDER 2: PAPER 5: GO SUB 9
990
7431 PLOT 40,56: DRAW 0,70: DRAW
 75,48: DRAW 75,-48: DRAW 0,-70:
 PLOT 80,56: DRAW 0,60: DRAW 40,
0: DRAW 0,-60: PLOT 80,56: DRAW
30,10: DRAW 0,50
7432 FOR f=8 TO 24 STEP 8: PLOT
80+f,116: DRAW 0,-(60-(f/4))+3:
NEXT f
7433 FOR f=10 TO 50 STEP 10: PLO
T 80,56+f: DRAW 30,10: NEXT f
7434 PLOT 130,56: DRAW 0,80: PLO
T 140,56: DRAW 0,80: FOR f=10 TO
 70 STEP 10: PLOT 130,56+f: DRAW
 10,0: NEXT f: PLOT 145,130: DRA
W -30,0: DRAW 0,20: DRAW 30,0: D
RAW 0,-20
7435 LET WW=1: LET NN=WW: LET UU
=WW: LET SS=WW: LET Q$="You are
outside a barn made of  wood": R
ETURN
7440 GO SUB 7246: GO SUB 7266
7445 LET nn=1: LET ss=NN: LET ee
=NN: LET q$="It is still dark bu
t you can    make out a faint pa
th": RETURN
7455 GO SUB 7246: LET ee=1: LET
ww=1: RETURN
7465 LET nn=1: LET ee=NN: GO SUB
 7246: RETURN
7470 BORDER 0: PAPER 0: GO SUB 9
990: INK 7: CIRCLE 128,116,59
7475 LET Q$="The tunnel is darke
r now and youcannot even see the
 walls.......": GO TO 3100
7480 BORDER 0: PAPER 0: GO SUB 9
990: CIRCLE 128,116,59: PLOT 128
,116
7485 LET SS=1: LET q$="The tunne
l continues.A spot of  light far
 away raises your hopes": RETURN

7490 BORDER 0: PAPER 0: GO SUB 9
990: CIRCLE 128,116,59: FOR f=1
TO 5: CIRCLE 128,116,f: CIRCLE 1
29,116,f: NEXT f
7495 LET nn=1: LET ss=NN: LET Q$
="The tunnel still goes on, but
  now the light is much nearer":
 RETURN
7500 BORDER 4: PAPER 4: GO SUB 9
990: LET i=6: GO SUB 9950: INK 0
: PLOT 50,56: DRAW 205,60,-PI/2:
 PLOT 1,120: DRAW 75,-30,-PI/4
7505 LET ee=1: LET q$="You come
out of the tunnel and  are dazzl
ed by the sun": RETURN
7510 GO SUB 7688: PLOT 0,100: DR
AW 150,10,-PI/5: DRAW 100,5,-PI/
6
7515 LET q$="Grass goes on for m
iles to the  north so you decide
 to go back": LET ss=1: RETURN
7520 GO SUB 7406: PLOT 0,80: DRA
W 50,-10: DRAW 75,5: DRAW 125,-1
0
7525 LET q$="A grassy bank leads
 up to a     forest": LET nn=1:
LET ee=1: RETURN
7530 GO SUB 7246: LET q$="It is
a little lighter here": LET ss=1
: RETURN
7545 GO SUB 7246: LET nn=1: LET
ss=1: RETURN
7555 GO TO 7545
7560 GO SUB 7530: LET ww=1: RETU
RN
7565 GO SUB 7530: LET ww=1: RETU
RN
7575 GO SUB 7250: LET ss=0: RETU
RN
7580 BORDER 4: PAPER 5: GO SUB 9
990: LET i=6: GO SUB 9950: INK 4
7581 PRINT AT 6,1;"
  ";AT 6,11;"          ";
AT 6,21;"        "
7582 PRINT AT 7,0;"







                 "
7583 FOR f=1 TO 4: PLOT 0,122-f:
 DRAW 80,0,2/-PI: DRAW 70,0,3/-P
I: DRAW 105,0,1.6/-PI: NEXT f
7584 GO SUB 7588
7585 LET nn=1: LET q$="You get o
ut of the boat and lookaround,Yo
ur view from the        mountain
looks down onto the    castle":
RETURN
7588 PRINT AT 7,16; PAPER 4; INK
 2;"          ";AT 8,16;"
        ";AT 9,16;"
     ";AT 10,16;"
 ";AT 10,20; PAPER 0;" ";AT 9,2
0;" ": RETURN
7590 BORDER 4: PAPER 5: GO SUB 9
990: LET i=6: GO SUB 9950: INK 4
: PLOT 0,170: DRAW 40,-40,2/PI:
INK 1: DRAW 170,0: INK 4: DRAW 4
0,40,2/PI: INK 1
7591 FOR f=40 TO 210 STEP 10: `PL
OT f,130: DRAW -5,-70,1.5/-PI: N
EXT f: FOR f=50 TO 200 STEP 20:
PLOT f,130: DRAW (RND*20)-10,-(1
0+(RND*10)): PLOT f+10,57: DRAW
(RND*20)-10,20+(RND*10): PLOT f+
5,115: DRAW (RND*20)-10,-(20+(RN
D*10)): NEXT f
7595 LET ee=1: LET q$="Your way
is blocked by a giant  waterfall
.": RETURN
7600 BORDER 7: PAPER 5: GO SUB 9
990: LET i=6: GO SUB 9950: INK 1
: GO SUB 7606
7601 GO SUB 7608
7602 PLOT 75,89: DRAW 180,0: PLO
T 24,89: DRAW -24,0
7605 LET ee=1: LET q$="You are i
n the midst of a quiet woodmans
village.The houses are of a stra
nge type made of bamboosticks":
RETURN
7606 PLOT 75,56: DRAW 0,40: DRAW
 10,0: DRAW -70,0: DRAW 35,30: D
RAW 35,-30: DRAW 0,-1: DRAW -70,
0: DRAW 10,0: DRAW 0,-39: PLOT 4
0,56: DRAW 0,20: DRAW 20,0,-PI:
DRAW 0,-20: RETURN
7608 FOR f=100 TO 220 STEP 40: P
LOT f,90: DRAW 0,15: DRAW -3,0:
DRAW 10,10: DRAW 10,-10: DRAW -1
6,0: DRAW 13,0: DRAW 0,-15: PLOT
 f+5,90: DRAW 0,5: DRAW 4,0,-PI:
 DRAW 0,-5: NEXT f: RETURN
7620 GO SUB 7960
7625 LET Q$="The woodland path l
eads straightahead": LET NN=0: L
ET ee=1: RETURN
7630 GO SUB 7406
7635 LET q$="You are on the outs
kirts of a  forest": LET ee=1:
LET nn=1: RETURN
7640 GO SUB 7930: PLOT 128,100:
DRAW 0,-44: PLOT 42,90: DRAW 0,-
34:: PLOT 84,97: DRAW 0,-41: PLO
T 213,90: DRAW 0,-34: PLOT 171,9
7: DRAW 0,-41
7645 PRINT AT 15,0;"You fall dow
n a primitive but   effective an
imal trap and land  on a stake":
 GO TO 9500
7650 GO SUB 7406
7655 GO SUB 7758: LET nn=1: LET
ss=1: RETURN
7660 GO SUB 7406
7665 LET q$="The trees are begin
ning to thin out and you can see
 a path ahead": LET ss=1: LET ee
=1: RETURN
7675 GO SUB 7930+u: LET q$="The
forest ends and you clearly see
a path to the south": RETURN
7680 GO SUB 7688
7681 PLOT 0,56: DRAW 255,50: DRA
W -70,-50:
7685 LET Q$="The path bends righ
t": LET ee=1: LET ss=1: RETURN
7688 BORDER 4: PAPER 4: GO SUB 9
990: LET I=6: GO SUB 9950: INK 0
: RETURN
7690 GO SUB 7809
7695 LET SS=1: LET EE=SS: LET WW
=SS: LET NN=SS: LET q$="The rive
r is shallower and       offers t
he chance to cross to a path on
the other side": RETURN
7700 BORDER 4: PAPER 4: GO SUB 9
990: LET i=6: GO SUB 9950: INK 0
: PLOT 0,130: DRAW 75,0,-2/PI/2:
 DRAW 80,10,2/-PI: DRAW 95,-5,3/
-PI
7701 INK 5: FOR f=1 TO 15: PLOT
0,70+f: DRAW 100,5,2/PI: DRAW 15
0,-10,2/-PI: NEXT f
7705 LET ee=1: LET ww=1: LET nn=
1: LET q$="The river grows bigge
r and      bigger as you walk to
wards it": RETURN
7715 GO SUB 7930+U: LET Q$="You
are in a forest clearing": LET n
n=0: RETURN
7720 GO SUB 7406: GO SUB 7681
7725 LET q$="The woodland path t
urns right": LET ss=1: LET nn=1:
 LET ww=1: LET ee=0: RETURN
7730 GO SUB 7406: GO SUB 7781
7735 LET q$="The woodland path t
urns left": LET ee=1: LET ss=1:
RETURN
7745 GO TO 7630+U
7750 GO SUB 7406
7755 LET WW=1: LET ee=1: GO SUB
7758: RETURN
7758 LET q$="This deep in the fo
rest there   are no paths, only
trees": RETURN
7760 GO SUB 7930
7765 LET SS=0: LET WW=1: LET q$=
"In a clearing you find a chest
 lying on the floor": LET nn=1:
LET ee=1: RETURN
7775 GO TO 7680+u
7780 PAPER 4: GO SUB 9990
7781 PLOT 255,56: DRAW -255,50:
DRAW 100,-50
7785 LET q$="The path bends left
": LET nn=1: LET ww=1: RETURN
7790 BORDER 1: PAPER 1: GO SUB 9
990: LET i=6: GO SUB 9950:
7791 INK 5: PLOT 50,56: DRAW 78,
50,2/PI: PLOT 80,56: DRAW 48,50,
2/PI: PLOT 110,56: DRAW 18,50,2/
PI: PLOT 146,56: DRAW -18,50,-2/
PI: PLOT 176,56: DRAW -48,50,-2/
PI: PLOT 206,56: DRAW -78,50,-2/
PI
7795 INK 0: PAPER 7: PRINT AT 15
,0;"You attempt to wade the rive
r...but half-way through the cur
rantgets too strong and you are
     swept away to your end.": GO
 TO 9500
7800 GO SUB 7809
7801 INK 0: IF d(9,1)=7 AND d(9,
2)=10 AND d(9,4)=0 THEN  PLOT 75
,90: DRAW 100,-10,PI/2: DRAW -10
0,10: DRAW 90,2: DRAW 10,-12: FO
R f=1 TO 5: PLOT 75+(f*10),90-f:
 DRAW 5,-(10+(f*2)+2): NEXT f: P
LOT 75,89: DRAW 100,-10,PI/2: FO
R f=9 TO 6 STEP -1: PLOT 75+(f*1
0),90-f: DRAW 5,-(50-(f*5)): NEX
T f
7802 IF d(9,1)=13 AND d(9,2)=9 A
ND d(9,4)=0 THEN  FOR f=1 TO 100
 STEP 10: PLOT 75+f,90-(f/10): D
RAW (90-f)+(f/10),0: NEXT f: PLO
T 174,82: PLOT 140,91: DRAW 25,0
: DRAW 2,-4: DRAW 1,0
7805 LET EE=1: LET WW=1: LET q$=
"You are by the side of a fast
 river, To the east there is a
 great castle": RETURN
7809 LET i=6: PAPER 5: GO SUB 99
90: GO SUB 9950: INK 1: FOR f=1
TO 8: PLOT 0,100+f: DRAW 70,-5:
DRAW 60,0: DRAW 60,10: DRAW 65,-
```

SPECTRUM PROGRAMMING

```
3: NEXT f: RETURN
7810 GO SUB 7206
7815 LET Q$="A massive tree stan
ds before you": LET ss=1: RETURN

7825 GO SUB 7720+u: LET ee=1: LE
T ww=0: RETURN
7835 GO SUB 7730+u: LET ee=0: RE
TURN
7845 GO SUB 7630+u: LET ww=1: RE
TURN
7850 GO SUB 7406: GO SUB 7606
7855 LET q$="You are in a cleari
ng with a    house": LET ww=1: L
ET ss=1: RETURN
7860 GO SUB 7391
7865 LET q$="You see a clearing
through the  trees": LET ss=1: L
ET ww=1: RETURN
7875 GO SUB 7960+U: LET Q$=Q$+"
into a curtain which is blocking
 the way east": LET ss=0: RETURN

7880 PAPER 4: INK 0: GO SUB 9990
7885 LET dd=1: LET q$="Green lea
ves surround you and   you can o
nly go down": RETURN
7890 BORDER 0: PAPER 0: GO SUB 9
990: RESTORE 7890
7891 PLOT 40,110: DRAW 0,-40: DR
AW 2,-2: DRAW 2,2: DRAW 0,39: DR
AW 60,-10: DRAW 0,-40: DRAW 2,-2
: DRAW 2,2: DRAW 0,40: DRAW 60,1
0: DRAW 0,-40: DRAW 2,-2: DRAW 2
,2: DRAW 0,40: DRAW -65,15: DRAW
 -68,-15
7892 FOR f=40 TO 100 STEP 5: PLO
T f,110-((f-40)/5): DRAW 0,-5: N
EXT f: FOR f=110 TO 165 STEP 5:
PLOT f,97+((f-105)/5): DRAW 0,-5
: NEXT f
7893 FOR g=1 TO 10: FOR f=1 TO 7
: INK f: PLOT 105,105: DRAW 10,1
5: DRAW -10,15: DRAW -10,-15: DR
AW 10,-15: DRAW 0,30: PLOT 115,1
20: DRAW -20,0: NEXT f: NEXT g
7894 FOR f=1 TO 7: READ n,m: BEE
P n,m: NEXT f
7895 PRINT AT 15,0;"You have rea
ched your aim.......But you aren
't safe yet.You set the crystal
to explode.You must now return t
o your own time": GO TO 4000
7899 DATA .2,5,.2,5,.2,5,.2,10,.
2,10,.2,10,1,20
7900 GO SUB 7206
7905 LET uu=1: LET ww=1: LET q$=
"You are under a giant tree": RE
TURN
7925 GO SUB 7720+u: LET NN=0: RE
TURN
7930 GO SUB 7406: PLOT 0,80: DRA
W 255,0,-PI/5
7935 LET q$="In the forest clear
ing you find the skeleton of a w
arrior": LET nn=1: LET ee=1: LET
 ss=1: RETURN
7940 GO SUB 7845: RETURN
7955 GO SUB 7780+u: LET ss=1: RE
TURN
7960 GO SUB 7688: PLOT 0,56: DRA
W 128,60: DRAW 127,-60
7965 LET q$="The path leads stra
ight on": LET ss=1: LET nn=1: LE
T ww=1: RETURN
7970 GO SUB 7680: PLOT 100,77: D
RAW -100,40: DRAW 60,-50:
7975 LET q$="A fork in the path
leads north": LET EE=1: LET WW=1
: LET NN=1: LET SS=0: RETURN
7985 LET q$="You are in the King
's orchard": LET ee=1: RETURN
7990 PAPER 0: GO SUB 9990: PLOT
70,56: DRAW -30,70,PI/4: DRAW -3
0,-70,PI/4: CIRCLE 60,80,2
7995 LET SS=1: GO SUB 7997: LET
q$=q$+" west here": RETURN
7997 LET q$="There is nothing bu
t a locked   door on the ": RETU
RN
8000 PAPER 0: GO SUB 9990: PLOT
240,56: DRAW -30,70,PI/4: DRAW -
30,-70,PI/4: CIRCLE 230,80,2
8005 LET NN=1: GO SUB 7997: LET
q$=q$+" east here": RETURN
8025 GO TO 7510+u
8030 GO SUB 7688: PLOT 0,100: DR
AW 250,-20,-PI/4: PLOT 255,130:
DRAW -150,-15,PI/3
8035 GO SUB 8058: RETURN
8040 GO SUB 7688: PLOT 0,100: DR
AW 128,30,-PI/5: DRAW 127,10,-PI
/4: GO SUB 7588
8045 GO SUB 8058: RETURN :
8050 GO SUB 7688: PLOT 0,80: DRA
W 255,-24,-PI/5
8055 GO SUB 8058: RETURN
8058 LET ss=1: LET nn=1: LET ww=
1: LET q$="To the south the gras
sy banks   still go on and on":
RETURN
8060 GO SUB 8068
8065 LET ss=0: GO SUB 8058: RETU
RN
8068 GO SUB 7688: PLOT 40,56: DR
AW 0,50: DRAW 150,0: DRAW -75,40
: DRAW -75,-40: PLOT 190,56: DRA
W 0,50
8069 PLOT 60,56: DRAW 0,30: DRAW
 20,0: DRAW 0,-30: RETURN
8075 GO SUB 7680+u: LET ee=0: LE
T ww=1: RETURN
8080 BORDER 2: INK 2: PAPER 4: G
O SUB 9990:
8081 PLOT 35,56: DRAW 0,100: DRA
W -5,0: DRAW 35,0: DRAW -5,0: DR
AW 0,-30: FOR f=60 TO 190 STEP 2
0: PLOT f,126: DRAW 10,0: DRAW 0
,10: DRAW 10,0: DRAW 0,-10: NEXT
 f
8082 DRAW 10,0: DRAW 0,30: DRAW
-5,0: DRAW 35,0: DRAW -5,0: DRAW
 0,-100: PLOT 30,156: DRAW 0,4:
FOR f=30 TO 60 STEP 10: PLOT f,1
60: DRAW 0,5: DRAW 5,0: DRAW 0,-
5: DRAW 5,0: NEXT f: OVER 1: PLO
T 70,160: DRAW -5,0: OVER 0: PLO
T 65,160: DRAW 0,-4
8083 PLOT 205,156: DRAW 0,4: FOR
 f=205 TO 235 STEP 10: PLOT f,16
0: DRAW 0,5: DRAW 5,0: DRAW 0,-5
: DRAW 5,0: NEXT f: OVER 1: PLOT
 245,160: DRAW -5,0: OVER 0: PLO
T 240,160: DRAW 0,-4
8084 FOR f=73 TO 203 STEP 20: PL
OT f,105: DRAW 0,10: DRAW 5,0,-P
I: DRAW 0,-10: DRAW -5,0: NEXT f
: PLOT 110,56: DRAW 0,25: DRAW 3
6,0,-PI: DRAW 0,-25: FOR f=115 T
O 128 STEP 5: PLOT f,56: DRAW 0,
((f/10)*8)-55: NEXT f: OVER 1: P
LOT 125,99: PLOT 125,100: PLOT 1
25,101: OVER 0: GO TO 8086
8085 LET nn=1: LET ww=1: LET ss=
1: LET q$="You are standing outs
ide the    King's palace": RETUR
N
8086 FOR f=130 TO 140 STEP 5: PL
OT 270-f,56: DRAW 0,((f/10)*8)-6
6: NEXT f: OVER 1: PLOT 130,100:
 PLOT 130,101: PLOT 130,102: OVE
R 0
8089 GO TO 8085
8090 GO SUB 8098
8091 INK 0: FOR f=39 TO 149 STEP
 10: FOR g=1 TO 3: PLOT f+g,95:
DRAW 5,-5: DRAW 5,5: NEXT g: NEX
T f
8092 FOR f=42 TO 162 STEP 29: PL
OT f,105: DRAW -2,0: DRAW 0,2: D
RAW 2,0: DRAW 0,5: DRAW -5,5: DR
AW 12,0: DRAW -5,-5: DRAW 0,-5:
DRAW 2,0: DRAW 0,-2: DRAW -4,0:
NEXT f
8093 PLOT 200,90: DRAW 0,-25: DR
AW -1,0: DRAW 0,25: DRAW -25,10:
 DRAW 0,-25: DRAW -1,0: DRAW 0,2
5: DRAW 25,10: DRAW 0,30: DRAW 2
5,-10,-PI: DRAW 0,-55: DRAW -1,0
: DRAW 0,25: DRAW -25,10: DRAW 2
5,-10: DRAW -25,-10
8095 LET NN=1: LET q$="You are i
nside the King's palacein the ba
nqueting hall.Seated ona tall ch
air there is a grand   old man":
 RETURN
8098 BORDER 3: PAPER 2: GO SUB 9
990: INK 0: PRINT AT 9,5;"
              ": FOR f
=50 TO 145 STEP 90: FOR g=1 TO 8
: PLOT f+g,100: DRAW 0,-40: NEXT
 g: NEXT f: FOR f=1 TO 3: PLOT f
+159,96: DRAW 0,7: NEXT f: RETUR
N
8100 PAPER 0: GO SUB 9990
8105 LET ww=1: LET q$="The room
is dark but you can     smell a b
ody": RETURN
8500 LET U=0: LOAD "" DATA D():
LOAD "" DATA E(): LOAD "" DATA F
(): LOAD "" DATA D$(): LOAD "" D
ATA E$(): LET X=F(1): LET Y=F(2)
: LET ST=F(3): GO TO 442
8870 GO SUB 7688: PLOT 0,56: DRA
W 128,60: DRAW 127,-60
8875 LET q$="The path leads stra
ight east": LET nn=1: LET ee=1:
LET ww=1: RETURN
8950 GO SUB 7780+u
9500 PAUSE 0: INK 0: PAPER 7: CL
S : BEEP 1,-10: BEEP .5,0: BEEP
.25,-15: BEEP 1,-25: PRINT "Your
 adventure ends here": GO TO 405
0
9800 DATA 10,9,7,1,2,0,4,9,0,5,3
,0,7,1,0,5,6,0,1,10,0,10,6,0,3,9
,0,10,10,4
9810 DATA 1,7,1,0,4,2,1,0,3,8,0,
0,10,10,1,0,1,3,1,0,9,3,0,0,10,8
,1,0,5,8,0,0,7,10,0,0,5,10,1,0,1
,3,5,0,7,6,0,0,4,2,2,0,2,2,0,0,8
,7,0,0,9,9,0,0,9,10,0,0,5,10,10,
0,1,7,1,0,2,3,0,0
9830 DATA "KING","BEAR","GOBLIN"
,"GOBLIN","WOOD-ELF","DWARF","EL
F","EAGLE","TROLL","GUARDIAN"
9840 DATA "SWORD","FOOD","ROPE",
"KEY","RING","BONE","FOOD","FISH
","BOAT","LANTERN","CUPBOARD","C
HEST","CHEST","CHEST","CURTAIN",
"DOOR","DOOR","DOOR","DOOR","DOO
R"
9850 DATA "KILL","","ASK","OPEN"
,"CLOSE","","TAKE","DROP","GIVE"
,"EAT","","EXAMINE"
9890 LET z$=("EAST," AND ee=1)+(
"WEST," AND ww=1)+("NORTH," AND
nn=1)+("SOUTH," AND ss=1)+("UP,"
 AND uu=1)+("DOWN," AND dd=1): P
RINT '"exits-";z$: RETURN
9900 LET R=0: FOR n=1 TO 10: IF
e(n,1)=x AND e(n,2)=y THEN  INPU
T INKEY$="y": PRINT TAB 9;"the "
;e$(n): IF E$(N)<>"DEAD-BODY" TH
EN  LET R=N
9910 NEXT n
9920 FOR m=1 TO 10: IF d(m,1)=X
AND D(M,2)=Y AND D(M,4)=0 AND D(
M,3)=0 THEN  INPUT INKEY$="y": P
RINT TAB 9;"the ";D$(M)
9925 NEXT M
9930 FOR m=11 TO 20: IF d(m,1)=X
 AND D(M,2)=Y THEN  INPUT INKEY$
="y": PRINT TAB 9;"the ";D$(M)
9935 NEXT M: INPUT INKEY$="Y": P
RINT TAB 9;"nothing special"
9940 RETURN
9950 INK 1: CIRCLE 40,160,10
9955 FOR n=1 TO 8: PLOT 31,155+n
: DRAW 18,0: NEXT n
9960 FOR n=1 TO 5: PLOT 31+n+1,1
63+n: DRAW 18-(n*2),0: NEXT n
9965 FOR n=1 TO 5: PLOT 31+n+1,1
56-n: DRAW 18-(n*2),0: NEXT n
9970 PLOT 34,153: PLOT 38,169: D
RAW 5,0
9975 PLOT 34,167: PLOT 33,165: D
RAW 5,5
9980 PLOT 33,166: PLOT 32,164
9985 RETURN
9990 FOR n=0 TO 14: INK 9: PRINT
 AT n,0;"
          ": NEXT n: RETURN
9999 LET nn=SIN PI: LET ss=NN: L
ET ee=NN: LET dd=NN: LET ww=NN:
LET uu=NN: RETURN
```

KINGDOM OF KULL

pull down MENUS

This article is all about pull-down menus. Specifically, about how you can use them in BASIC (with the aid of a morsel of machine code of course). Before I start, let's begin with some definitions.

A *menu* is something you normally get in restaurants, which lists the different choices of food available to you. Computers, however, can't eat anything except chips. For this reason computer menus are allowed to list not only food, but in fact anything that it's possible to list at all. When a menu appears on the screen, what you see is a list of choices. Usually this list is headed with a title. The computer will then wait until you're ready to order. You make your selection and then, once the computer has carried out whatever tasks it's been told to, the menu will disappear from the screen. Whatever was *underneath* the menu will reappear and the screen will be as it was before the menu appeared.

Nests

Sometimes you see menus happening in *nests*. When this happens lots of birds start chirping for worms and things. On the screen you may witness one menu or information panel appearing on top of a previous menu. Furthermore, you may then get even more menus layering themselves on top of



Toni Baker shows how to give your programs that 'state of the art' look, with this routine for creating pull-down menus.

these. When all tasks are complete you will see the menus disappearing, precisely in reverse order — each time leaving the screen exactly as it was before the particular menu appeared. When the final (ie first) menu disappears the screen is as it was to begin with.

This suggests some sort of STACK, because the first menu in becomes the last menu out. We are familiar with at least three stacks already — the machine stack, the calculator stack, and the GOSUB stack. If we wish to create yet another kind of stack still, we must first of all decide where to put it.

What I propose to do in this article is to provide a comprehensive machine code program with access points to make most of its functions available to BASIC. If this is done then it means that nested menus and information panels may be created and recalled using only simple BASIC statements.

Essentially, the menu idea is a very simple one. PUSHing a menu consists of (1) storing the contents of the screen portion which lies underneath the proposed menu position; (2) printing a list, with a title and a pretty border, at the designated position; and (3) providing some means of allowing the user to choose one of the options.

Alternatively we may wish to use step (1) only, or steps (1) and (2) only, to allow for a variety of types of information panel, as well as just the standard kind of menu.

The reverse process, that of POPping a menu, has only one step; the contents of the screen underneath the menu must be restored. This will cause the menu or information panel to disappear from the screen.

Obscure RAM

For this program, I have decided to use a little-known and seldom-used region of RAM in which to store the stack of panel information. It is possible to store

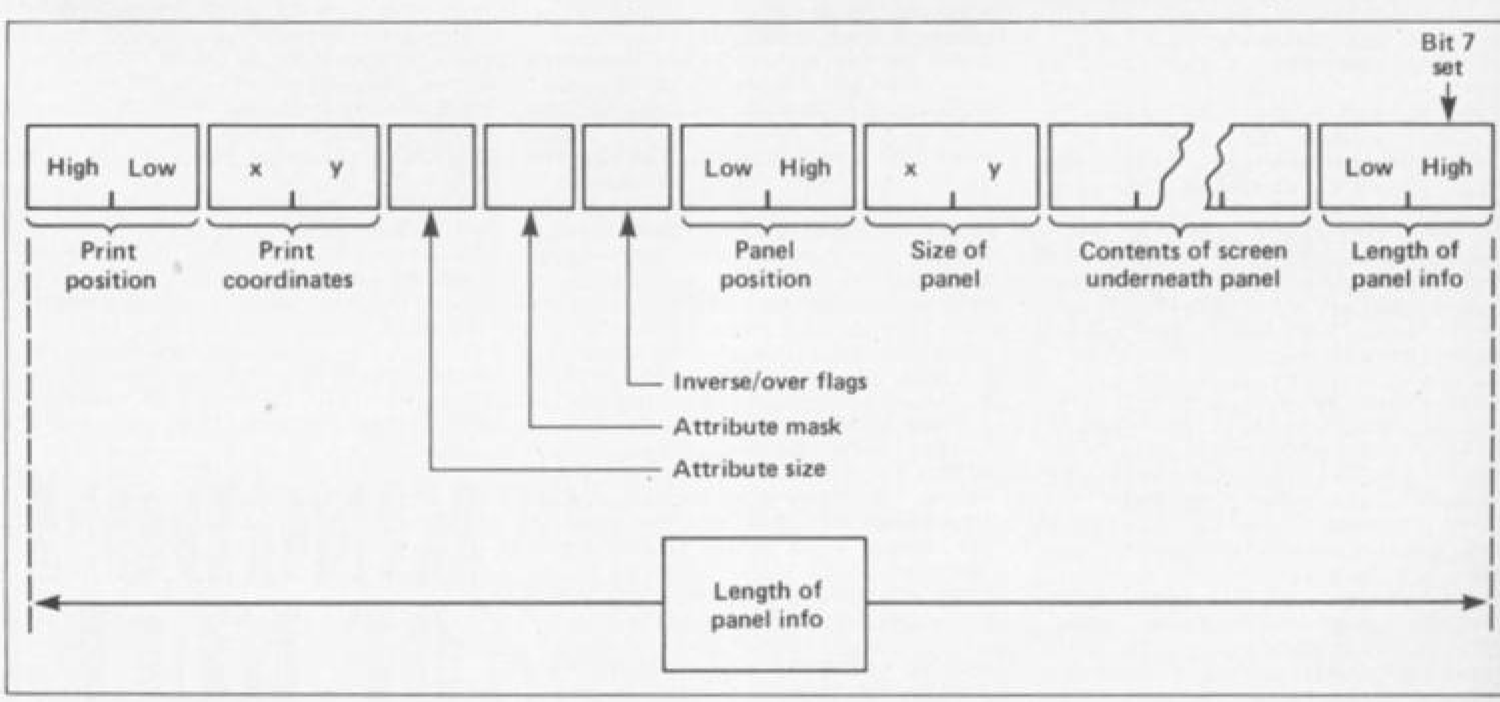


Figure 1. Stack entry diagram.

information BETWEEN the BASIC program and the variables area. This region has the following advantages: (1) It is impossible for the stack to grow too big — error 4, *Out of memory* will occur if too much memory is asked for; (2) it will not overwrite any machine code (or in fact anything); and (3) the information will not be erased by the ROM between BASIC statements (as the workspace would be).

My new stack grows upwards. Figure one shows a single stack entry diagramatically. Note that each stack entry may be of variable length. As you can see from the diagram the print position and attribute colours are also stacked, so that when the menu disappears printing may continue as normal.

The first two bytes of information store the address on the screen of the current print position. Note that this address is stored HIGH BYTE FIRST. This is unusual in Spectrum machine code, but is done for a good reason. Remember that to the left (in the diagram) of the panel information is the BASIC program itself. The RUN and LIST commands must know where to stop — it would not do if the panel information somehow got confused with program. The first byte beyond the BASIC program must be in the range 40 to FF, so as to distinguish it from a program line. Putting the high byte first ensures this.

Notice also that the very last byte of panel information has bit 7 set. This too is for a reason. The machine code program must be able to tell whether or not the panel stack is empty. If, directly below the variables area, it finds a byte with bit 7 set, then the stack is non-empty. Note that if the stack WERE in fact empty then the byte directly below the variables area would be OD (enter) which of course has bit seven reset. Once the principle of the panel information stack is understood, the rest of the algorithm is boringly straight-forward. There's a lot of it, but this is because there is lots of work to do — not because of any untoward complexity.

To clear menu stack entirely:
RANDOMIZE USR MCLEAR

To create information panel:
RANDOMIZE USR PANEL: PRINT " [control ;] " height , width

To create menu panel (without choice):
RANDOMISE USR MPANEL: PRINT " [control ;] " string "|, string|"

To create menu panel, and allow choice of options:
LET variable = USR MENU: PRINT " [control ;] " string|" , string|"

To undo a menu or information panel:
RANDOMIZE USR MENUOFF
or:
LET variable = USR MENUOFF
(The latter case will assign the variable with 0 if the menu stack is now empty, or with 1 otherwise).

"control" may be either: AT y-coordinate, x-coordinate
PAPER colour
INK colour
BRIGHT status
FLASH status

" [...] " means "optional, but may be repeated"

" |...| " means "at least one must be included, more repeats are optional"

Figure 2. New commands for manipulating menus.

Back to BASIC

Let's look at the BASIC now. How does it operate? Figure two lists the five new statements which are allowed. Note that the middle three have rather complicated syntaxes. To begin with each of the three statements is in two parts, separated by a colon. The second part always begins "PRINT" — but this is *not* a PRINT statement — it just looks like one. You may follow the word PRINT by an optional number of controls, which may specify the colour of the menu (eg PAPER 6); or the position of the menu (eg AT8,8;). In the case of USR PANEL you then have to specify the size of the panel — first the height, then the width, separated by a comma. In the remaining two cases (USR MPANEL and USR MENU) you have to supply a list of strings. The first one is the title, and the rest are the choices available on the menu.

You don't have to specify the menu size once the strings are listed. The machine code will work that out for itself (ie height = number of strings + 1; width = maximum length of string + 2). Although the syntax looks complicated at first glance, in fact it's remarkably easy once you're used to it.

Figure Three is an example BASIC program. It doesn't do very much, apart from show off the menus. Each feature is demonstrated at least once. Try it — it's very interesting.

You can use the BASIC statements in your own programs. As a guide — you can CREATE a menu with the **LET X = USR MENU:** statement, followed by "PRINT" and a list of strings. You don't have to use X of course — any variable will do. From then on, in the BASIC program, this menu will be on the screen, and X will be assigned with 1 if the first item on the list was chosen, 2 if the second item on the list was chosen, and so on. To remove this menu from the screen you should use **RANDOMIZE USR MENUOFF**. Alternatively, it may be the case that X has been assigned with zero — if this is so it means that the menu has been abandoned, and has already been removed from the screen.

From a user's point of view, when you are confronted with a menu you must manipulate the bar cursor to the appropriate choice using the UP and DOWN cursor keys, and then press ENTER when the cursor is in the right place. Alternatively, to abandon the menu altogether, just press DELETE.

Well, that's all from me for this article. All this talk about menus has made me hungry. Excuse me while I make myself a microfych supper.

```
1 LET MCLEAR = 32959
2 LET PANEL = 33460
3 LET MPANEL = 33465
4 LET MENU = 33467
5 LET MENUOFF = 33595
6 PAPER 7: INK 0: BORDER 7
7 RANDOMIZE USR MCLEAR
10 LET X = USR MENU: PRINT BRIGHT 1;"MENU","MICROCHIPS","MICROEGGS",
  "MICROBEANS","MICROTOMATOES"
20 IF X=0 THEN STOP
30 GO SUB 100*X
40 RANDOMIZE USR MENUOFF
50 GO TO 10
100 LET Y = USR MENU: PRINT AT X,4; PAPER 6; "MONTHS","JANUARY","FEBRUARY",
  "MARCH","APRIL","MAY","JUNE","JULY","AUGUST","SEPTEMBER","OCTOBER",
  "NOVEMBER","DECEMBER"
110 IF Y THEN RANDOMIZE USR MENUOFF
120 RETURN
200 RANDOMIZE USR MPANEL: PRINT AT X,4; PAPER 6; "IMPORTANT MESSAGE","MAKE TEA NOW"
210 PAUSE 0: PAUSE 0
220 RANDOMIZE USR MENUOFF
230 RETURN
300 RANDOMIZE USR PANEL: PRINT PAPER 6; 5,20
310 FOR I = 1 TO 5
320 PRINT "twenty spaces"
3330 NEXT I
340 PLOT 0,175
350 DRAW 159,0: DRAW 0,-23: DRAW -159,0: DRAW 0,23
360 PRINT AT 1,1; FLASH 1; "DON'T PANIC!"
370 PAUSE 0: PAUSE 0
380 RANDOMIZE USR MENUOFF
390 RETURN
```

Figure 3. Example Basic program.

```
400 LET Y = USR MENU: PRINT AT X,4; PAPER 6; "RODENTS","RAT","HAMSTER","GERBIL"
410 IF Y = 0 THEN RETURN
420 IF Y = 2 THEN LET Z = USR MENU: PRINT AT X+Y,8; PAPER 5; "YEAH","ISN'T","THIS",
   "FUN": IF Z THEN RANDOMIZE USR MENUOFF: PAUSE 0: PAUSE 0
430 RANDOMIZE USR MENUOFF
440 RETURN
2000 CLEAR 32767
2010 LOAD "menu" CODE
2020 GO TO 1
```

```
                        ORG 8000
90909090  MENU_UDGS     DEFB 90 90 90 90
90909090                DEFB 90 90 90 90       Definition for graphic-A.
09090909                DEFB 09 09 09 09
09090909                DEFB 09 09 09 09       Definition for graphic-B.
90909090                DEFB 90 90 90 90
9F8080FF                DEFB 9F 80 80 FF       Definition for graphic-C.
09090909                DEFB 09 09 09 09
F90101FF                DEFB F9 01 01 FF       Definition for graphic-D.
00000000                DEFB 00 00 00 00
FF0000FF                DEFB FF 00 00 FF       Definition for graphic-E.
0000      PANEL_LEN     DEFW 0000
0000      PANEL_ADDR    DEFW 0000
00        PANEL_TYPE    DEFB 00
00        STRING_COUNT  DEFB 00
0000      MENU_AT       DEFW 0000              System variables used by program.
          ____________________
                        ORG 8030
7C        ATTR_ADDR     LD A,H                 A:= high part of screen address.
E638                    AND 38                 Identify screen third.
1F                      RRA
1F                      RRA
1F                      RRA                    A:= screen third number.
F658                    OR 58
67                      LD H,A                 HL:= corresponding attribute address.
C9                      RET
          ____________________
                        ORG 803A
F5        DOWN_1        PUSH AF
24                      INC H                  Presume within character square.
7C                      LD A,H
E607                    AND 07
200A                    JR NZ,DOWN_1_EXIT      Jump if this is the case.
7D                      LD A,L
C620                    ADD A,20
6F                      LD L,A                 Presume crossing screen thirds.
3804                    JR C,DOWN_1_EXIT       Jump if this is the case.
7C                      LD A,H
D608                    SUB 08
67                      LD H,A                 Must be within screen third.
F1        DOWN_1_EXIT   POP AF
C9                      RET                    Return.
          ____________________
                        ORG 804D
2A4B5C    UNSTK_PANEL   LD HL,(VARS)           HL: points to variables area.
2B                      DEC HL                 HL: points to top of panel stack.
CB7E                    BIT 7,(HL)
CBBE                    RES 7,(HL)
C8                      RET Z                  Return if panel stack empty.
46                      LD B,(HL)
2B                      DEC HL
4E                      LD C,(HL)              BC:= length of panel information.
C5                      PUSH BC                Stack length of panel information.
23                      INC HL
23                      INC HL                 HL: points to variables area.
A7                      AND A
ED42                    SBC HL,BC              HL: points to panel information.
E5                      PUSH HL                Stack this address.
56                      LD D,(HL)
23                      INC HL
5E                      LD E,(HL)              DE:= previous print position.
23                      INC HL
ED53845C                LD (DF_CC),DE          Restore previous print position.
5E                      LD E,(HL)
23                      INC HL
56                      LD D,(HL)              DE:= previous print coordinates.
23                      INC HL
ED53885C                LD (S_POSN),DE         Restore previous print coordinates.
5E                      LD E,(HL)              E:= previous attribute byte.
23                      INC HL
56                      LD D,(HL)              D:= previous attribute mask.
23                      INC HL
ED538D5C                LD (ATTR_P),DE         Restore previous attribute colours
ED538F5C                LD (ATTR_T),DE          and transparent mask.
7E                      LD A,(HL)              A:= previous INVERSE/OVER status.
23                      INC HL
32915C                  LD (P_FLAG),A          Restore INVERSE/OVER status.
5E                      LD E,(HL)
23                      INC HL
56                      LD D,(HL)              DE:= address within screen of panel.
23                      INC HL
4E                      LD C,(HL)              C:= width of panel, in squares.
23                      INC HL
0600                    LD B,00                BC:= width of panel, in squares.
7E                      LD A,(HL)              A:= height of panel, in squares.
23                      INC HL                 HL: points to previous data from scr.
F5                      PUSH AF                Stack height of panel in squares.
87                      ADD A,A
87                      ADD A,A
87                      ADD A,A                A:= height of panel in pixels.
D5                      PUSH DE                Stack panel address within screen.
D5        U_P_LOOP_1    PUSH DE                Stack address of pixel row to restore.
C5                      PUSH BC                Stack width of panel.
EDB0                    LDIR                   Restore one pixel row of panel.
C1                      POP BC                 BC:= width of panel.
D1                      POP DE                 DE:= address of row just restored.
EB                      EX DE,HL
CD3A80                  CALL 803A,DOWN_1
EB                      EX DE,HL               DE:= address of next row to restore.
3D                      DEC A
20F2                    JR NZ,U_P_LOOP_1       Restore all pixels to panel.
D1                      POP DE                 DE:= address of first row.
EB                      EX DE,HL
CD3080                  CALL 8030,ATTR_ADDR
EB                      EX DE,HL               DE:= address of first attribute row.
F1                      POP AF                 A:= height of panel in squares.
C5        U_P_LOOP_2    PUSH BC                Stack width of panel.
D5                      PUSH DE                Stack address of attribute row.
EDB0                    LDIR                   Restore next attribute row.
E3                      EX (SP),HL             HL:= addr of attr row just restored.
012000                  LD BC,0020
09                      ADD HL,BC              HL:= address of next attribute row.
EB                      EX DE,HL               DE:= address of next attribute row.
E1                      POP HL                 HL: points into panel information.
C1                      POP BC                 BC:= width of panel.
3D                      DEC A
20F1                    JR NZ,U_P_LOOP_2       Restore all attributes to panel.
E1                      POP HL                 HL: points to start of panel info.
C1                      POP BC                 BC:= length of panel information.
CDE819                  CALL 19E8,RECLAIM_2    Reclaim memory used by panel info.
224B5C                  LD (VARS),HL           Restore variables-area pointer.
F6FF                    OR FF                  Reset the zero flag.
C9                      RET                    Return.
          ____________________
                        ORG 80BF
CD4D80    MENU_CLEAR    CALL UNSTK_PANEL       Unstack topmost panel, if one exists.
20FB                    JR NZ,MENU_CLEAR       Try again if there may be more.
C36B0D                  JP 0D6B,CLS            Clear the screen, and return.

                        ORG 80C7
F5        NEW_PANEL     PUSH AF                Stack required attribute colours.
78                      LD A,B                 A:= required height of panel.
82                      ADD A,D                A:= maximum line number of panel.
380C                    JR C,NP_ERROR          Jump if greater than 255d.
FE19                    CP 19
3008                    JR NC,NP_ERROR         Jump if height off of screen.
79                      LD A,C                 A:= required width of panel.
83                      ADD A,E                A:= maximum column number of panel.
3804                    JR C,NP_ERROR          Jump if greater than 255d.
FE21                    CP 21
3802                    JR C,NEW_PANEL_2       Jump unless width is off of screen.
CF04      NP_ERROR      RST 08/DEFB 04         Report 5 - Out of screen.
C5        NEW_PANEL_2   PUSH BC                Stack size of intended panel.
D5                      PUSH DE                Stack position of intended panel.
110900                  LD DE,0009             DE:= no of bytes needed per chr sqr.
62                      LD H,D
68                      LD L,B                 HL:= height of intended panel.
CDF82A                  CALL 2AF8,MULT_HL_DE   HL:= HL * DE
59                      LD E,C                 DE:= width of intended panel.
CDF82A                  CALL 2AF8,MULT_HL_DE   HL:= number of bytes needed for
                                               screen information for whole panel.
1E0D                    LD E,0D
19                      ADD HL,DE              HL:= length of required panel info.
22B880                  LD (PANEL_LEN),HL      Store this length.
44                      LD B,H
4D                      LD C,L                 BC:= length of required panel info.
2A535C                  LD HL,(PROG)           HL: points to start of program area.
E5                      PUSH HL                Stack this address.
2A4B5C                  LD HL,(VARS)           HL: points to start of variables area.
22BA80                  LD (PANEL_ADDR),HL     Store this as address of panel info.
2B                      DEC HL                 HL: points to top of panel stack.
C5                      PUSH BC
CD5516                  CALL 1655,MAKE_ROOM    Make room for panel information.
C1                      POP BC
EB                      EX DE,HL               HL: points to penultimate byte
                                               of new room.
71                      LD (HL),C
23                      INC HL
70                      LD (HL),B              Store length of panel info.
CBFE                    SET 7,(HL)             Set bit 7 to signal "stack not empty".
E1                      POP HL                 HL:= address of program area.
22535C                  LD (PROG),HL           Restore system variable.
2ABA80                  LD HL,(PANEL_ADDR)     HL:= address of new room.
ED5B845C                LD DE,(DF_CC)          DE:= current print position.
```

```
72                  LD (HL),D
23                  INC HL
73                  LD (HL),E          Store in panel info.
23                  INC HL
ED5B885C            LD DE,(S_POSN)     DE:= current print coordinates.
73                  LD (HL),E
23                  INC HL
72                  LD (HL),D          Store in panel information.
23                  INC HL
ED5B8D5C            LD DE,(ATTR_P)     DE:= current attribute byte & mask.
73                  LD (HL),E
23                  INC HL
72                  LD (HL),D          Store in panel information.
23                  INC HL
3A915C              LD A,(P_FLAG)      A:= current INVERSE/OVER status.
77                  LD (HL),A          Store in panel information.
23                  INC HL
3E02                LD A,02
E5                  PUSH HL
CD0116              CALL 1601,CHAN_OPEN   Select stream 2 (the screen).
E1                  POP HL
C1                  POP BC             BC:= position of panel.
E5                  PUSH HL            Stack pointer into panel info.
CD9B0A              CALL 0A9B,AT_B_C   PRINT AT B,C;
EB                  EX DE,HL           DE:= address within screen of panel.
E1                  POP HL             HL: points into panel info.
73                  LD (HL),E
23                  INC HL
72                  LD (HL),D          Store screen addr in panel info.
23                  INC HL
C1                  POP BC             BC:= size of panel.
71                  LD (HL),C
23                  INC HL
70                  LD (HL),B          Store in panel information.
23                  INC HL
EB                  EX DE,HL           DE:= address at which to store data
                                       from screen; HL:= screen address.
F1                  POP AF             A:= required attribute byte.
E5                  PUSH HL
6F                  LD L,A
2600                LD H,00
228D5C              LD (ATTR_P),HL
228F5C              LD (ATTR_T),HL     Set required colours.
FD7457              LD (P_FLAG),H      Also set OVER 0; INVERSE 0;
E1                  POP HL             HL:= screen address of panel.
78                  LD A,B             A:= height of panel in squares.
0600                LD B,00            BC:= width of panel in squares.
F5                  PUSH AF            Stack height of panel in squares.
87                  ADD A,A
87                  ADD A,A
87                  ADD A,A            A:= height of panel in pixels.
E5                  PUSH HL            Stack address of first row.
E5    N_P_LOOP_1    PUSH HL            Stack address of row.
C5                  PUSH BC
EDB0                LDIR               Store one row of panel.
C1                  POP BC
E1                  POP HL             HL:= address of row just stored.
CD3A80              CALL 803A,DOWN_1   HL:= address of next row to store.
3D                  DEC A
20F4                JR NZ,N_P_LOOP_1   Store all screen information for panel
E1                  POP HL             HL:= address of first row.
CD3080              CALL 8030,ATTR_ADDR   HL:= address of first attribute row.
F1                  POP AF             A:= height of panel in squares.
C5    N_P_LOOP_2    PUSH BC            Stack width of window.
E5                  PUSH HL            Stack address of attribute row.
EDB0                LDIR               Store one attribute row.
E1                  POP HL             HL:= addr of attr row just stored.
012000              LD BC,0020
09                  ADD HL,BC          HL:= address of next attribute row.
C1                  POP BC             BC:= width of window.
3D                  DEC A
20F3                JR NZ,N_P_LOOP_2   Store all attribute info for panel.
C9                  RET                Return.
```

```
                    ORG 8178
C5    MENU_NEWLINE  PUSH BC
14                  INC D              Increment y coordinate.
D5                  PUSH DE
42                  LD B,D             B:= required y coordinate.
4B                  LD C,E             C:= required x coordinate.
CD9B0A              CALL 0A9B,AT_B_C   PRINT AT B,C;
D1                  POP DE
C1                  POP BC
C9                  RET                Return.
```

```
                    ORG 8183
C5    MENU_STRING   PUSH BC
D5                  PUSH DE
2A5F5C              LD HL,(X_PTR)      HL: points to reqd string parameters.
23                  INC HL
5E                  LD E,(HL)
23                  INC HL
56                  LD D,(HL)          DE:= original address of string.
23                  INC HL
46                  LD B,(HL)          B:= length of string.
23                  INC HL
23                  INC HL
225F5C              LD (X_PTR),HL      (X_PTR): pts to nxt string params.
2A655C              LD HL,(STKEND)     HL: pts to end of dynamic RAM.
ED52                SBC HL,DE
380C                JR C,M_STRING_2    Jump if address after dynamic RAM.
2A2A80              LD HL,(PANEL_ADDR) HL:= addr at which info was inserted.
ED52                SBC HL,DE
3005                JR NC,M_STRING_2   Jump if address below panel info.
2A2880              LD HL,(PANEL_LEN)  HL:= length of panel information.
19                  ADD HL,DE
EB                  EX DE,HL           DE:= new address of string.
04    M_STRING_2    INC B              B:= length of string, plus one.
1804                JR MS_CHR_NEXT     Jump forward.
1A    MS_CHR_LOOP   LD A,(DE)          A:= next chr from string.
13                  INC DE             DE: points to next chr.
D7                  RST 10             Print the character.
0D                  DEC C              Decrement number of squares left.
10FA  MS_CHR_NEXT   DJNZ MS_CHR_LOOP   Print whole of string.
0C                  INC C              C:= number of squares left, plus one.
1803                JR MS_SPC_NEXT     Jump forward.
3E20  MS_SPC_LOOP   LD A,"space"
D7                  RST 10             Print a space.
0D    MS_SPC_NEXT   DEC C
20FA                JR NZ,MS_SPC_LOOP  Print all remaining spaces.
D1                  POP DE
C1                  POP BC
C9                  RET
```

```
                    ORG 81BB
F5    NEW_MPANEL    PUSH AF            Stack required attribute byte.
D5                  PUSH DE            Stack position of panel.
78                  LD A,B             A:= number of strings supplied.
F5                  PUSH AF            Stack number of strings.
2A655C              LD HL,(STKEND)     HL: points to end of calculator stk.
110000              LD DE,0000         DE:= smallest number possible.
2B    NMP_LOOP      DEC HL
46                  LD B,(HL)
2B                  DEC HL
4E                  LD C,(HL)          BC:= length of string.
2B                  DEC HL
2B                  DEC HL
2B                  DEC HL
EB                  EX DE,HL           HL:= length of longest string so far.
A7                  AND A
ED42                SBC HL,BC
09                  ADD HL,BC
3002                JR NC,NMP_STRLEN   Jump if HL greater than BC.
60                  LD H,B
69                  LD L,C             HL:= length of longest string so far.
EB    NMP_STRLEN    EX DE,HL           HL: points into calculator stack.
3D                  DEC A
20EC                JR NZ,NMP_LOOP     Scan all strings.
A2                  AND D
2005                JR NZ,NMP_ERROR    Error if length gtr than 255d.
4B                  LD C,E             C:= maximum string length.
CB79                BIT 7,C
2802                JR Z,NMP_PANEL     Jump if length less than 128d.
CF04  NMP_ERROR     RST 08/DEFB 04     Report 5, Out of screen.
225F5C NMP_PANEL    LD (X_PTR),HL      Store pointer to string parameters.
F1                  POP AF
47                  LD B,A             B:= number of strings
D1                  POP DE             DE:= intended position for menu.
F1                  POP AF             A:= required attribute byte.
C5                  PUSH BC
D5                  PUSH DE
04                  INC B              Allow one extra line for border.
0C                  INC C
0C                  INC C              Allow two extra columns for border.
CDC780              CALL 80C7,NEW_PANEL   Assign panel information area.
D1                  POP DE
C1                  POP BC
2A5F5C              LD HL,(X_PTR)      HL: points to first string parameter.
22655C              LD (STKEND),HL     Empty calculator stack from here on.
3E14                LD A,14
D7                  RST 10             PRINT INVERSE ...
3E01                LD A,01
D7                  RST 10             1;
3E20                LD A,"space"
D7                  RST 10             Print a space.
CD8381              CALL 8183,MENU_STRING   Print the menu title.
3E20                LD A,"space"
D7                  RST 10             Print a space.
3E14                LD A,14
D7                  RST 10             PRINT INVERSE ...
AF                  XOR A
D7                  RST 10             0;
2A7B5C              LD HL,(UDG)
E5                  PUSH HL            Stack user defined graphics address.
```

```
210080                LD HL,MENU_UDGS
227B5C                LD (UDG),HL              Use specified UDG's.
05                    DEC B                    B:= number of remaining strings.
C5                    PUSH BC
CD7881    MMP_PRINT   CALL 8178,MENU_NEWLINE   Move print position to next line
                                               of menu panel.
3E90                  LD A,"graphic-A"
D7                    RST 10                   print left hand border.
CD8381                CALL 8183,MENU_STRING    Print next string.
3E91                  LD A,"graphic-B"
D7                    RST 10                   Print right hand border.
10F2                  DJNZ MMP_PRINT           Print all string.s.
FD362600              LD (X_PTR)hi,00          Clear pointer.
CD7881                CALL 8178,MENU_NEWLINE   Move print position to last line
                                               of menu panel.
3E92                  LD A,"graphic-C"
D7                    RST 10                   Print bottom left hand corner.
3E94      MMP_UNDER   LD A,"graphic-E"
D7                    RST 10                   Print underside border.
0D                    DEC C
20FA                  JR NZ,MMP_UNDER          Print whole of underside border.
3E93                  LD A,"graphic-D"
D7                    RST 10                   Print bottom right hand corner.
C1                    POP BC                   B:= number of strings, excluding
                                               title; C:= maximum length of string.
E1                    POP HL                   HL:= original address of UDG's.
227B5C                LD (UDG),HL              Restore system variable.
C9                    RET                      Return
          ________________________________________

                      ORG 8241
D5        NEW_MENU    PUSH DE
CDBB81                CALL 81BB,NEW_MPANEL     Print menu on screen.
D1                    POP DE
0C                    INC C
0C                    INC C                    C:= full width, including borders.
3E01                  LD A,01                  A:= initial menu choice.
F5        M_M_LOOP    PUSH AF                  Stack menu choice.
82                    ADD A,D                  A:= screen line number of menu choice.
6F                    LD L,A
AF                    XOR A                    A:= 00.
67                    LD H,A                   HL= scr line no of menu choice.
29                    ADD HL,HL
29                    ADD HL,HL
29                    ADD HL,HL
29                    ADD HL,HL
29                    ADD HL,HL
D5                    PUSH DE
57                    LD D,A                   DE:= x coordinate of panel.
19                    ADD HL,DE
110058                LD DE,5800
19                    ADD HL,DE                HL:= attribute address of menu choice.
D1                    POP DE
C5                    PUSH BC
7E        M_M_MARK    LD A,(HL)                A:= attribute byte from row.
EE10                  XOR 10                   Change paper colour.
77                    LD (HL),A                Make change on screen.
23                    INC HL                   HL: points to next attr byte in row.
0D                    DEC C
20F8                  JR NZ,M_M_MARK           Mark entire row.
C1                    POP BC
FDCB01AE  M_M_WAIT    RES 5,(FLAGS)            Signal "Ready for a new key".
FDCB016E  M_M_WAIT_2  BIT 5,(FLAGS)
28FA                  JR Z,M_M_WAIT_2          Wait till key pressed.
3A085C                LD A,(LAST_K)            A:= chr code of key pressed.
D60A                  SUB 0A
2009                  JR NZ,M_M_KEY_B          Jump unless key is "cursor down".
F1                    POP AF                   A:= current menu choice.
B8                    CP B
3C                    INC A                    Move one line down.
380C                  JR C,M_M_NEXT            Jump unless below last entry.
3E01                  LD A,01                  Go to first entry.
1808                  JR M_M_NEXT              Jump forward.
3D        M_M_KEY_B   DEC A
2013                  JR NZ,M_M_KEY_C          Jump unless key is "cursor up".
F1                    POP AF                   A:= current menu choice.
3D                    DEC A                    Move one line up.
2001                  JR NZ,M_M_NEXT           Jump unless above first entry.
78                    LD A,B                   Go to last entry.
F5        M_M_NEXT    PUSH AF                  Stack new menu choice.
C5                    PUSH BC
2B        M_M_UNMARK  DEC HL                   HL: points to next attr byte in row.
7E                    LD A,(HL)                A:= attribute byte from row.
EE10                  XOR 10                   Restore paper colour.
77                    LD (HL),A                Make change on screen.
0D                    DEC C
20F8                  JR NZ,M_M_UNMARK         Remove marker entirely.
C1                    POP BC
F1                    POP AF                   A:= menu choice.
18B4                  JR M_M_LOOP              Loop back for next try.
3D        M_M_KEY_C   DEC A
2006                  JR NZ,M_M_KEY_D          Jump unless key is "delete".
F1                    POP AF                   A:= menu choice.
CD4D80                CALL 804D,UNSTK_PANEL    Remove menu from screen.
AF                    XOR A                    A:= 00.
C9                    RET                      Return, with A containing zero.
3D        M_M_KEY_D   DEC A
20C4                  JR NZ,M_M_WAIT           Jump back unless key is "enter".
F1                    POP AF                   A:= menu choice.
C9                    RET                      Return, with A containing choice.
          ________________________________________

                      ORG 82A4
DF        SEPARATOR?  RST 18                   A:= current character.
FE27                  CP "'"
C8                    RET Z                    Set zero flag for "'".
FE2C                  CP ","
C8                    RET Z                    Set zero flag for ",".
FE3B                  CP ";"
C9                    RET                      Or set zero flag for ";".
          ________________________________________

                      ORG 82AE
CDA482    SEPARATOR   CALL 82A4,SEPARATOR?     Test current character.
C8                    RET Z                    Return with valid separators.
CF0B      ERROR_C     RST 08/DEFB 0B           Report C, Nonsense in BASIC.
          ________________________________________

                      ORG 82B4
AF        B_PANEL     XOR A                    A:= 00.
1806                  JR B_INTERPRET
3E01      B_MPANEL    LD A,01
1802                  JR B_INTERPRET
3E03      B_MENU      LD A,03
210000    B_INTERPRET LD HL,0000
222B80                LD (MENU_AT),HL          Initialise menu location.
6F                    LD L,A
222C80                LD (PANEL_TYPE),HL       Define panel type and reset
                                               string count.
3E02                  LD A,02
CD0116                CALL 1601,CHAN_OPEN      Select stream 2 (the screen).
CD4D0D                CALL 0D4D,TEMPS          Use permanent colours.
DF                    RST 18                   A:= current character from BASIC line.
FE3A                  CP ":"
20E8                  JR NZ,ERROR_C            Error unless chr is ":".
E7                    RST 20                   A:= next chr from BASIC line.
FEF5                  CP "PRINT"
2029                  JR NZ,ERROR_C            Error unless chr is "PRINT".
FD340D                INC (SUBPPC)             Increment statement count.
E7        BI_LOOP     RST 20                   A:= next chr from BASIC line.
FEAC                  CP "AT"
2010                  JR NZ,BI_COLOURS         Jump unless chr is "AT".
CD791C                CALL 1C79,NEXT_2NUM      Evaluate AT parameters.
CD0723                CALL 2307,STK_TO_BC      C,B:= AT parameters.
61                    LD H,C
68                    LD L,B
222B80                LD (MENU_AT),HL          Store required AT coordinates.
CDAE82                CALL 82AE,SEPARATOR      Expect a separator.
18E8                  JR BI_LOOP               Jump back to continue interpretting.
CDF221    BI_COLOURS  CALL 21F2,CO_TEMP_3      Evaluate colour commands.
3805                  JR C,BI_MENU             Jump if no colour commands present.
CDAE82                CALL 82AE,SEPARATOR      Expect a separator.
18E1                  JR BI_LOOP               Jump back to continue interpretting.
212C80    BI_MENU     LD HL,PANEL_TYPE
CB46                  BIT 0,(HL)
280F                  JR Z,BI_PANEL            Jump if panel only requested.
CD8C1C    BI_STR_LOOP CALL 1C8C,EXPT_STRING    Expect a string expression.
212D80                LD HL,STRING_COUNT
34                    INC (HL)                 Increment number of strings counted.
CDA482                CALL SEPARATOR?          Is there a separator?
2016                  JR NZ,BI_MENU_GO         Jump if not.
E7                    RST 20                   Skip over the separator.
18F1                  JR BI_STR_LOOP           Loop back to count remaining strings.
CD7A1C    BI_PANEL    CALL 1C7A,EXPT_2NUM      Evaluate panel size.
CD0723                CALL 2307,STK_TO_BC      C,B:= panel size.
78                    LD A,B                   A:= panel width.
41                    LD B,C                   B:= panel height.
4F                    LD C,A                   C:= panel width.
ED5B2B80              LD DE,(MENU_AT)          DE:= panel position.
3A8F5C                LD A,(ATTR_T)            A:= colours required.
C3C780                JP 80C7,NEW_PANEL        Jump to create panel.
ED4B2C80  BI_MENU_GO  LD BC,(PANEL_TYPE)       B:= number of strings.
ED5B2B80              LD DE,(MENU_AT)          DE:= panel position.
3A8F5C                LD A,(ATTR_T)            A:= colours required.
CB49                  BIT 1,C
CABB81                JP Z,81BB,NEW_MPANEL     Jump to create menu panel, if reqd.
CD4182                CALL 8241,NEW_MENU       Else create menu panel and choose.
0600                  LD B,00
4F                    LD C,A                   BC:= menu choice.
C9                    RET                      Return.
          ________________________________________

                      ORG 833B
CD4D80    MENU_OFF    CALL 804D,UNSTK_PANEL    Unstack topmost panel, if it exists.
2A4B5C                LD HL,(VARS)             HL: points to variables area.
2B                    DEC HL                   HL: points to top of panel stack.
010000                LD BC,0000
CB7E                  BIT 7,(HL)
C8                    RET Z                    Return if no panels left on stack.
03                    INC BC                   BC:= one.
C9                    RET                      Return.
```

CROSSFIRE

Details of the newly founded National Computer Club in this month's mailbag.

Nationwide Club

Perhaps ZX readers would be interested to know about the recent formation of The National Computer Club. Many magazines have moved towards game playing and away from real computing. Unfortunately, localised computer clubs have, all too often, done the same thing. Consequently, computer hobbyists have tended to drift away and their expertise has gone with them. This has left large numbers of enthusiasts without the personal contact with other enthusiasts that progression and development of ideas needs.

The National Computer Club (NCC) is seeking to fill the gap by providing all the benefits of a local computing club but on a nationwide scale, giving members the benefit of personal contact with a large pool of knowledge, expertise and experience.

Our aim is to produce an environment, within the club, where members can contact one another to find solutions to problems, answers to queries or to form computing relationships with other, like-minded, members.

The club caters for all levels of expertise from beginners to experts and for all machines — even home built. This is important because we view computing as 'computing' and not 'Spectruming', 'Commodore 64ing' or even 'Beebing'. No disrespect intended since most of us limit our computing to just one machine, but interests, within the overall term 'computing', range from Basic programming right through to machine building and a great deal of information and programming (with modifications) is applicable to all machines, e.g. address decoding, machine code flow charts, etc.

The NCC is a 'computing' club and therefore caters for all computing interests. Of course a Spectrum user, thinking of buying a particular program, might want to hear the views of others who already have the program. In the NCC he can ask them. I should add that the excellent hobby of games writing is a part of the NCC, but games playing, on its own, is not. However, players who wish to move into computing are very welcome.

The way that members make contact with others, whether for help or for computing relationships, is through our monthly bulletin which is for the free use of members. Included in it are: sales, wants, queries (could be difficulties or general interest), general projects, items of interest such as utility routines, techniques, etc. and whatever else the members would like to say. For instance somebody might need the pin-out of a particular logic chip. He could ask for it in the bulletin and other members who just happen to have that information would send it directly to him. He might also have stated what he is doing, etc and asked for others who are doing the same to get in touch for the sharing of information and mutual progression.

Finally, for a mere SAE, I would be pleased to send further details of the NCC to any reader of ZX.

Contact: Philip Craven, NCC, 212 Dudley Hill Road, Bradford BD2 3DF.

Calculating

Toni Baker's machine code calculator series has been very enlightening, even though the references given to locations in the ROM do not apply to my T/S 2068.

I am writing because there is a slight error in the part 4 of the series. Everything that has been said about the machine code calculator, or other ROM routines, is accurate for the 2068 as well — after adjusting for the different locations of the routines. However, the author on page 64 (October) suggests that it is not possible to define the factorial function with a BASIC DEF FN statement. Well, the Spectrum or the 2068 is a machine of many surprises, so perhaps we should not be too shocked to discover that indeed we can use DEF FN to define the factorial function using only BASIC.

The technique used for this is called a recursive function definition.

The following statement will define a function whose value — if the argument is a positive integer — is the factorial of x.

```
10 DEF FM F(N)=N*VAL(("FN F(N-1)-1"
AND N>1)+"+1") OR N   1
```

Similarly, it is also possible to define the function F$ that he mentions in only BASIC, or a function which does what INSTR does on other computers. These would look as follows:

```
20 DEF FN F$(X$,X)=(X$ AND
X>=.5)+VAL$ (("FN F$(X$,X-1)" AND
X>=1.5)+"+""""")
30 DEF FM I(S,A$,B$)=S*(A$(S TOS+LEN
B$-1)=B$)+VAL (("FN I(S+1,A$,B$)"
AND S+LEN B$ <=LEN A$(S TO S+LEN
B$-1)<>B$)+"O")
```

For F$ and F I suggest you see part 4 of the machine code calculator article. The function FN I(S,A$,B$) has as its value the location of the first occurence of B$ in A$ after the Sth character. For example, FN I(3,"ZX Computing", "put") is 7, since the string "put" can be found starting with the 7th character in "ZX Computing". If I had set the number at 8 instead of 3, the answer would have been O since "put" does not occur starting with or after the eighth character.

This is not to say that defining such functions in BASIC is the way to go — the function definitions above are pretty slow if some of the numbers or strings are large. This is because finding the factorial of 10 using the above function for it actually has to evaluate the function 10 times. Also, one must be careful with recursive functions to make sure that the function will reach an end eventually. Finally, because of the way the Spectrum and 2068 handle recursive functions, the function call must be the last thing evaluated.

Keep up the good work with your fine magazine.

Steven V Gunhouse, Winsor, Ontario, Canada.

Pen Pals

I own a Spectrum+ with Microdrive and I wish to contact other Spectrum owners to exchange information and programs.

Natner Naji, Cairo Q, 8-33-307, Baghdad, Iraq.

Rejuvenating Your Ribbon

Printer ribbons seem to cost a small fortune these days, so I decided to try the services of a firm called ALADDINK, who re-ink fabric ribbons. The prices vary, my Epson RX80 was £2.05, which I think was about their top price, a saving of over £5.50 on the price of a new one. My ribbon was returned within a week, well inked, and accompanied by a personalized order form for my next one.

If you want to give them a try, they will ink any ribbon for £2.00, and send it back with a quote for doing the same make and model in the future.

Carol Brooksbank, Coventry.

Aladdink, 4 Hurkur Crescent, Eyemouth, Berwickshire TD14 5AP, Scotland.

Zebras

I have a Zebra disk system that I use on my TS2068, both of which have romswitches. I would like to learn more, or exchange ideas, about the Zebra (Portuguese) with other owners.

As far as I know, my disk interface romswitch may be the first of its kind and it makes changing from Spectrum to 2068 a snap. Would you please print this to help me find some interested Zebra owners.

Ken Diederich, 312 N.Bailey, Jacksonville, Arkansas 72076, U.S.A.

"All this memory saving for adventurers is all very well, but what about the rest of us!" I hear ZXC enthusiasts say. Well, here's a memory saving routine just for you ('though I suspect the adventurers may find a use for it as well). It's called "Fastfile", and is a system for holding information, in any form, in a DIMentioned string of 40,000 characters with a machine code routine, searching at about 50,000 characters per second to extract from it the information required and print to screen. Without further ado, let's get to work.

Type in the machine code loader, **Program 1,** RUN it, and enter the numbers from **Table A,** reading across the lines. As you enter each number it will be displayed so you can keep check. Note any mistakes, and correct them at the end with:
POKE address, correct number

Now NEW the machine — your code is safe above RAMTOP — and type in **Program 2.** This is the driver program and must be entered exactly as printed for reasons that will be obvious later. Note that the "STOP" in lines 10, 1010 and 2000 is the token, and is entered in symbol shift mode.

Remember, "Fools rush in where angels fear to tread", so polish up your halo, and SAVE the program and CODE, just in case of mistakes with:
CLEAR: SAVE "fastfile" LINE 9000: SAVE "FastCODE" CODE 65263,111 and VERIFY both parts.

Now you're ready to try it out. Type RUN (ENTER) and the menu will appear. The options are chosen by pressing the appropriate number (If you get an error message, and you may, as there is a minimum of error trapping to make as much memory as possible available for the file, restart with GOTO 100, never RUN). This is what you can do:

1. Entry

This adds an entry to the file, provided there is enough space (You're told how much space is free each time). The maximum

RANDOM MEMORY



INPUT length is about half a screen (Remember to restart with GOTO 100 if you get an Out of Memory message). If you want to save space, but avoid filling out the ends of lines with spaces to prevent word splitting, use the PRINT comma trick. For those who missed the earlier articles this is what you do. After typing the last character you want on the line get into E Mode, hold on to the Caps Shift key, and press 6, followed by 0. The cursor will jump to the next line (three of these in succession would leave a blank line within an entry at a cost of only three bytes!).

Make a file to experiment on using the fore and surnames of your family. After the last entry pressing just ENTER will return you to the menu.

2. Search

Select Option 2, and answer the "Key?" prompt by entering the word or phrase for which you wish to search. The machine code, which incidentally originated from the good old days of the ZX81, zips through your file, PRINTing out all entries which include that key. After each, the prompt "Erase?" will appear. Pressing "y" will erase that entry, "n" will continue the search. On completion the word END will be displayed. Pressing just ENTER will return you to the menu.

Try the following with your "names" file:—
a) Enter a forename — only that name will be displayed.
b) Try the surname — all entries with that surname will appear.
c) Try "Bloggs" (assuming that's not your name!) — just the END message will appear.
d) Try a single letter that you know is in the file — any entries which included that letter will appear **as many times as they contain the letter.** For example, John Jones would appear twice if the key were J o or n, but only once if s were entered. The moral of this exercise is that the

PROGRAM 1

```
10 CLEAR 65262: FOR f=65263 TO
65374: INPUT i: PRINT f,i: POKE
f,i: NEXT f
```

more specific the key used, the more selective the routine becomes. So if you were using the program as an index to magazine articles it would be better to reference Spectrum programs as sp1, rather than just sp, as in the latter case **any** entries where an s is followed by a p would be displayed.

3. Save

The whole BASIC program and the variables is SAVEd. Why not just the data array? Because you also need the values held in other variables, for example the file pointer, n. After VERIFYing you will be returned to the menu.

4. Load

Use this option to LOAD in an existing file for searching or updating. Existing files should only be LOADed in this way for interrogation. Don't be tempted to just LOAD in a SAVEd file directly as it will probably crash!

5. New

This clears the file by RUNning the program and resetting the arrays.

Clyde Bish presents a memory saving 'Fastfile' routine.

It would be nice to exlain how the machine code operates, but as usual space precludes that opportunity. Suffice it to say that the routine compares what is held in a$() in the VARS area with what is in b$(). It is therefore important that you make **no** alteration to line 10 until after DIM b$(40002) or these arrays will not be in the correct places in VARS for the routine to find them. If you alter the length of the program you will also need to reset v to a new value by PEEKing the VARS system variable using PRINT PEEK 23627 + 256 * PEEK 23628.

Fastfile (Microdrive)

And now, for microdrive owners, a version of "Fastfile" especially for you.

PROGRAM 2

```
  10 LET o=PI-PI: DIM a$(VAL "31"): DIM b$(VAL "40002"): LET l=SGN PI: LET b$(l)=" STOP ": LET n=VAL "2": LET h=VAL "100": LET v=VAL "24668": LET s1=VAL "23670": LET s2=s1+l
 100 CLS : PRINT "OPTIONS"''"1 ENTRY"''"2 SEARCH"''"3 SAVE "''"4 LOAD "''"5 NEW ": PAUSE o: CLS : GO TO (VAL INKEY$+VAL "1000"-h AND INKEY$<"6")+h
1000 CLS : PRINT VAL "40002"-n;" SPACES LEFT": INPUT "Entry"' LINE n$: IF n$="" THEN GO TO h
1010 LET n$=n$+" STOP ": IF LEN n$>VAL "40000"-n THEN PRINT "FILE OVERLOAD": PAUSE h: GO TO h
1015 LET b$(n TO n+LEN n$)=n$: LET n=n+LEN n$: GO TO VAL "1000"
2000 INPUT "Key?"' LINE c$: LET a$=c$+" STOP ": POKE VAL "65263",o: POKE VAL "65264",o
2110 LET p=USR VAL "65265": IF p>o AND p<n THEN GO TO VAL "2120"
2116 PRINT ''"END": PAUSE o: GO TO h
2120 LET p=p-l: IF b$(p)<>" STOP " THEN GO TO VAL "2120"
2124 LET s=p+l
2125 LET p=p+l: IF b$(p)<>" STOP " THEN PRINT b$(p);: GO TO VAL "2125"
2130 PRINT #o;"Erase?": PAUSE o: INPUT ;: IF INKEY$<>"y" THEN PRINT '': GO TO VAL "2110"
2140 RANDOMIZE (p+l+v+VAL "48"): POKE VAL "65363",PEEK s1: POKE VAL "65364",PEEK s2: RANDOMIZE (n-p): POKE VAL "65366",PEEK s1: POKE VAL "65367",PEEK s2: RANDOMIZE (s+v+VAL "48"): POKE VAL "65369",PEEK s1: POKE VAL "65370",PEEK s2: RANDOMIZE USR VAL "65362": LET n=n-(p+SGN h-s): PRINT '': GO TO VAL "2110"
3000 INPUT "filename?",n$: SAVE n$ LINE h: PRINT " VERIFY ": VERIFY "": GO TO h
4000 INPUT "filename?",n$: LOAD n$
5000 RUN
9000 CLEAR VAL "65262": LOAD *"m";1;"fastCODE"CODE : POKE VAL "23609",VAL "50": BORDER VAL "6": RUN
```

The program is essentially the same as the cassette version except that the Interface 1 ROM takes over much of the donkey work and, of course, speeds up the LOAD, SAVE and VERIFY routines.

The file array b$() is set to a length of 29000 characters. This length enables you to hold three files named a, b and c on a clear cartridge, plus the boot program (which sets everything running) and the machine code, giving a total storage of 87000 characters. It also allows for a more user-friendly program with single key-press controls, not to mention avoiding an encounter with the infamous Interface 1 ROM bug which switches on your microdrive permanently! (If this does ever happen **do not** power-down. You may lose data. Surprisingly, it is better to pull out the cartridge first whilst the motor is running!).

Type in the machine code as previously described, then NEW and enter **Program 3.** Now to prepare the cartridge. Format the cartridge as described in the manual, then RUN the program. It will stop with an error message. Fear not, saith he. All is well. Enter, as a command: **LET n$ = "a": GOTO 100**

The menu will appear. Choose 4, then press "a" in response to the "Filename?" prompt. The microdrive will run much longer than usual as it is trying to erase an, at the moment, non-existent file. When the menu reappears choose 4 again, and this time press "b". The third time the menu appears press 4 then "c".

Now BREAK out of the program, NEW the machine, and type in the boot, **Program 4.** Save the machine code still on board and the boot program with: **SAVE * "m";1;"run" LINE**

PROGRAM 3

```
  10 LET n=2: DIM a$(31): DIM b$(12): LET b$(1)=" STOP ": LET s1=23670: LET s2=s1+1
 100 CLS : LET s=2: PRINT "File ";n$''"OPTIONS"''"1 ENTRY"''"2 SEARCH (to Printer)"''"3 SEARCH (to Screen)"''"4 SAVE"''"5 LOAD"''"6 DELETE"
 110 PAUSE 0: CLS : LET c$=INKEY$: IF c$<"1" OR c$>"6" THEN GO TO 100
 120 GO TO VAL c$+1000
1000 CLS : PRINT 29002-n;" SPACES LEFT": INPUT "Entry? (ENTER for menu)"' LINE e$: IF e$="" THEN GO TO 100
1010 LET e$=e$+" STOP ": IF LEN e$>29002-n THEN PRINT "OVERLOAD": LET c$="o": GO SUB 4005: PRINT ''"Load new file then remake entry": GO TO 5000
1015 LET b$(n TO n+LEN e$)=e$: LET n=n+LEN e$: GO TO 1000
2000 LET s=3
3010 CLS : INPUT "Key?"' LINE c$: LET a$=c$+" STOP "
3020 POKE 65263,0: POKE 65264,0
3030 LET p=USR 65265: IF p>0 AND p<n THEN GO TO 3060
3040 PRINT ''"END OF FILE"''"LOAD NEW FILE?"''"Press a,b,c"''"(n for MENU)"''"(s to SAVE updated file)": PAUSE 0: LET c$=INKEY$: IF c$="n" THEN GO TO 100
3045 IF c$="s" THEN GO TO 4005
3048 LET n$=c$: GO TO 5010
3060 LET p=p-1: IF b$(p)<>" STOP " THEN GO TO 3060
3070 LET p=p+1: IF b$(p)<>" STOP " THEN PRINT #s;b$(p);: GO TO 3070
3080 PRINT #s'': IF INKEY$<>"" THEN PRINT '" STOP ": FOR f=1 TO 200: NEXT f: PAUSE 0: GO TO 100
3090 GO TO 3030
4000 PRINT "filename?": PAUSE 0: LET n$=INKEY$
4005 ERASE "m";1;n$: SAVE *"m";1;n$ LINE 100
4010 VERIFY *"m";1;n$: IF c$="s" THEN GO TO 5000
4015 IF c$="o" THEN RETURN
4020 GO TO 100
5000 PRINT ''"filename?": PAUSE 0: LET n$=INKEY$
5010 LOAD *"m";1;n$: GO TO 100
6000 CLS : INPUT "Key?"' LINE c$: LET a$=c$+" STOP ": POKE 65263,0: POKE 65264,0
6010 LET p=USR 65265: IF p>0 AND p<n THEN GO TO 6060
6020 PRINT ''"END": PAUSE 0: GO TO 100
6060 LET p=p-1: IF b$(p)<>" STOP " THEN GO TO 6060
6065 LET s=p+1
6070 LET p=p+1: IF b$(p)<>" STOP " THEN PRINT b$(p);: GO TO 6070
6080 PRINT #0;"Erase?": PAUSE 0: INPUT ;: IF INKEY$<>"y" THEN PRINT '': GO TO 6010
6100 LET v=PEEK 23627+256*PEEK 23628: RANDOMIZE (p+1+v+48): POKE 65363,PEEK s1: POKE 65364,PEEK s2: RANDOMIZE (n-p): POKE 65366,PEEK s1: POKE 65367,PEEK s2: RANDOMIZE (s+v+48): POKE 65369,PEEK s1: POKE 65370,PEEK s2: RANDOMIZE USR 65362: LET n=n-(p+1-s): PRINT '': GO TO 6010
```

Table A

0	0	221	42	75	92	17	12
0	221	25	42	75	92	17	44
0	25	94	35	86	235	237	91
239	254	167	237	82	68	77	42
75	92	25	17	49	0	25	221
86	0	122	237	177	234	35	255
237	67	239	254	201	221	229	229
221	35	221	86	0	122	254	226
32	22	42	75	92	17	49	0
25	235	225	221	225	167	237	82
34	239	254	237	75	239	254	201
190	32	3	35	24	218	225	221
225	24	196	33	148	96	1	4
0	17	145	96	237	176	201	

10: SAVE * "m";1;"fastCODE" CODE 65263, 111 (where "run" is the three letters, not the Keyword). VERIFY both parts and there you are.

To use the system, RANDOMIZE USR 0 to reset the machine, then press RUN (Enter). The boot will load, then LOAD in the code and file a (which is DIMentioned but empty).

The menu is essentially the same as before.

The ENTRY option will automatically SAVE a file when it is full, and invite you to LOAD in a new file to continue.

SEARCH gives a printer option (by altering the stream) as well as to the screen. The search can be stopped at any time by pressing a letter. A return is made to the menu by a second press. When the end of a file is reached there is the option to continue the search with another file (after SAVEing the existing one if updated). Alternatively a return can be made to the menu.

Obviously with SAVE the existing a, b or c file is erased before the update is SAVEd and VERIFYed.

As with save, the filename for LOAD is a single letter.

The DELETE option is now separate, but operates as before. The key is searched for with "y" to erase, "n" to leave — you may have more than one entry with that key — and Enter for the menu.

This routine will, of course, search through any information in b$() to locate a key and so could be used in any situation where fast retrieval is required. One such use would be vocabulary searching in an adventure (You see, I didn't leave you out!). Alternatively it could be used to search for words understood in an "Elisa"-type program it experiment with A.I.

I'll show you how it could be done and then you can improve on it.

Have a look at **Program 5** as you read what follows:—

Line **800**: reset variables and the arrays you've seen before.
830: R1 we need later. The program will stop if you've said "bye".
840: the user response goes into i$. A suitably cutting response is printed if this is a null string.
850: the input is printed to screen with a leading capital.
860-861: spaces in the input string are replaced by STOP tokens, plus one at the end. The counter n is set to string length + 1.
862: Now to business! The data is RESTOREd. Look at Line 900 for a moment. You'll notice a sequence of word, phrase, phrase triplets. In essence the routine takes each response word in turn and searches for a match in k$. If one is found (i.e. p > 0 AND p < n) then one of the phrases READ along with k$ and held in x$ and y$ is printed. Which, is determined by R1.
882: if the search falls through the loop, no match has been found as one of a sequence of general answers is given.
885: If too many general answers are given then a request is made to change the subject.
900: this is the first example of many DATA lines. You can produce the rest yourself but remember three things:—
a) the items must be in threes — keyword, phrase 1, phrase 2
b) the order of keywords in the data list is **most** important. Common words must be at the end with less likely ones at the beginning or the latter will never be found.
c) my original program had a vocabulary of 100 keywords. If you want more or less you must change the number ending the FOR statement in line 862:2

To end, here's an idea for an April Fool program to catch out an ardent Arcader (I know its early, but it'll take you that long to sort out the DATA!). Have a simple arcade game on board as well, with a timer line to switch control to your "chat" program after a given period (Use the FRAMES system variable). You could then have the machine do a fake "reset crash" (Use PAPER 0, then PAPER 7 with your own PRINTed copywrite notice) followed by a reincarnation and on-screen communicaton from the "Spirit of the Machine" offering to have a chat.

See who you can catch! Good fooling!

PROGRAM 4

```
 10 POKE 23609,30: BORDER 6: CL
EAR 65262: LOAD *"m";1;"fastCODE
"CODE : LOAD *"m";1;"a"
```

PROGRAM 5

```
 800 LET r=0: DIM a$(31): DIM b$
(64): LET b$(1)=" STOP ": LET k$
=""
 830 LET r1=RND: IF k$="goodbye"
 OR k$="bye" OR k$="cheerio" THE
N STOP
 840 INPUT PAPER 5;"Please talk
to me"' PAPER 7;"(No punctuation
 or capitals     except I)"'' LI
NE i$: IF i$="" THEN PRINT BRIGH
T 1'"Don't be stupid!": PAUSE 50
: GO TO 840
 850 PRINT BRIGHT 1''(CHR$ (CODE
 i$(1)-32) AND i$(1)>="a")+(CHR$
 CODE i$(1) AND i$(1)<"a");i$(2
TO )''
 860 FOR f=1 TO LEN i$: IF i$(f)
=" " THEN LET i$(f)=" STOP "
 861 NEXT f: LET b$(2 TO )=i$: L
ET n=LEN i$+1
 862 RESTORE : FOR f=1 TO 100: R
EAD k$,x$,y$: LET a$=k$+" STOP "
: POKE 65263,0: POKE 65264,0: LE
T p=USR 65265: IF p>0 AND p<n TH
EN PRINT PAPER 6;(x$ AND r1).5)+
(y$ AND r1<=.5)'': GO TO 830
 882 NEXT f: PRINT PAPER 6;("I s
ee" AND r=1)+("Tell me more" AND
 r=2)+("This is getting interest
ing" AND r=3)+("I'm not sure I u
nderstand" AND r=0): IF r<4 THEN
 LET r=r+1: GO TO 830
 885 PRINT PAPER 6;"This is bori
ng! Please change    the subject"
: LET r=0: GO TO 830
 900 DATA "said","Be careful of
what others say","Who cares!","s
ometimes","Only sometimes?","Not
 often?","occasionally","Try mor
e often","Only occasionally?","l
ike","I agree","In what way?","s
ame","I agree","How?","alike","E
xactly alike?","Really?","everyb
ody","Surely not!","Really every
one?","everyone","I don't believ
e that","Really?"
 901 REM Other DATA lines follow
```

STREAMS AND CHANNELS

The concluding part of Toni Baker's Windows program.

Last month we began experimenting with windows. The listing continues . . .

As a demonstration, the extra program WIND_DEMO which I've tagged on at the end at address B544 will open a window twenty-four squares wide by eight squares high, positioned AT 2,1 (relative to the whole screen) with yellow paper and blue ink. Furthermore, the window will be a SLOW window, so word's won't ever be cut in half, and although the standard charcter set is used, they are defined to be seven bits wide, not eight, so you get more characters than you would normally. Running the program once will open the window and attach it to stream four.

Thereafter PRINT 4 will print onto the window.

Follow through

For those of you who wish to follow the program through and understand how it all works, I'll tell you that the program starts running from location WINDOW (address B4F1) with the A register containing the character to be printed, whenever RST 10 is used with this channel.

Oh — incidentally — while we're talking about the WINDOW routine, take a look at the four instructions following the label WIND_CTRL. The CALL instruction carries out the control code function, the POP instruction restores the control character to the A register, and the RET instruction terminates everything — the routine has finished — control will then pass back to the RST 10 sub-routine itself, and then back to the PRINT statement which caused the RST 10 to be used. But . . . what's this AND A instruction doing just before the RET? The comment beside the instruction reads "Reset the carry flag". Why? — Surely everything's finished now. We shouldn't need to worry about flags should we?

Unfortunately we do. You see when a control code such as PAPER 4 or INVERSE 1 is used in a PRINT statement then the appropriate control codes are sent to RST 10 to be carried out. The PRINT routine expects the carry flag to be reset on return from such a routine, and will produce the error message "C Nonsense in BASIC" if this is not the case.

Next month I'll give you no less than two new channels: a modified network channel for owners of the ZX Interface 1 which will successfully communicate with a QL, and a channel which will allow users of the Spectrum 128 to use the standard ZX Printer whilst in 128K mode, saving a lot of money in the process. See you then.

The following subroutine prints a newline - ie it moves the print position to the left hand edge of the next line down.

```
                              ORG B2BB
CDC3B2     ENTER              CALL B2C3,ENTER_1          Print a single newline once.
DDCB0956                      BIT 2,(IX+W_FLAGS)
C8                            RET Z                      Return if using single height.
DD7E0E     ENTER_1            LD A,(IX+W_YCOORD)         A:= current y coordinate.
3C                            INC A                      A:= new y coordinate.
DDBE0F                        CP (IX+W_HEIGHT)
C2B1B1                        JP NZ,B1B1,LINE_A          Jump if in range to move print pos.
DDCB096E                      BIT 5,(IX+W_FLAGS)
2812                          JR Z,SCROLL                Jump if scroll pause disabled.
DD3515                        DEC (IX+W_SCROLLS)         Decrement scroll count.
200D                          JR NZ,SCROLL               Jump unless scroll pause required.
DD7E0F                        LD A,(IX+W_HEIGHT)
DD7715                        LD (IX+W_SCROLLS),A        Re-initialise scroll count.
3E7F       SCROLL_PAUSE       LD A,7F
DBFE                          IN A,(FE)                  Scan part of the keyboard.
1F                            RRA
38F9                          JR C,SCROLL_PAUSE          Pause until SPACE pressed.
CD541F     SCROLL             CALL 1F54,BREAK_KEY
D2000D                        JP NC,0D00,REPORT_D        Give error report if BREAK pressed.
DD7E0E                        LD A,(IX+W_YCOORD)         A:= y coordinate of bottom line.
F5                            PUSH AF
CDB1B1                        CALL B1B1,LINE_A           Move print pos to start of bottom
                                                         line of window.
C1                            POP BC                     B:= number of lines to copy.
CD06B2                        CALL B206,GET_WIDTH        C:= width of window in
DD6E12                        LD L,(IX+W_HOME)lo
DD6613                        LD H,(IX+W_HOME)hi         HL:= address of top left corner.
CD71B1                        CALL B171,PAGE_7           Use RAM page 7 in case screen 1 used.
C5         SCN_LOOP_1         PUSH BC                    Stack loop counter.
0600                          LD B,00                    BC:= width of line.
C5                            PUSH BC
E5                            PUSH HL
CDA5B1                        CALL B1A5,ATTR_ADDR        HL:= addr of this attribute line.
112000                        LD DE,0020
EB                            EX DE,HL                   DE:= addr of this attribute line.
19                            ADD HL,DE                  HL:= addr of next attribute line.
EDB0                          LDIR                       Copy one attribute line.
E1                            POP HL
C1                            POP BC
E5                            PUSH HL
CD9BB1                        CALL B19B,DOWN_8           HL:= address of next line.
D1                            POP DE                     DE:= address of this line.
E5                            PUSH HL
```

SPECTRUM PROGRAMMING

```
3E08                   LD A,08                 A:= number of rows per line.
C5         SCR_LOOP_2  PUSH BC
D5                     PUSH DE
E5                     PUSH HL
EDB0                   LDIR                    Copy one row from line.
E1                     POP HL
D1                     POP DE
C1                     POP BC
24                     INC H                   HL:= addr of next row of next line.
14                     INC D                   DE:= addr of next row of this line.
3D                     DEC A
20F3                   JR NZ,SCR_LOOP_2        Copy whole line.
E1                     POP HL
C1                     POP BC
10D6                   DJNZ SCR_LOOP_1         Transfer all required lines.
0B                     DEC BC                  BC:= length of line, less one.
DD7E14                 LD A,(IX+W_ATTR)        A:= attribute byte.
CDDEB1                 CALL B1DE,CLW_LINE      Clear bottom line.
C36CB1                 JP B16C,PAGE_0          Restore RAM page zero and return.
```

The following subroutine deals with all control codes except for comma-control and TAB (these are dealt with by CHR_TYPE_2 at address B12F). On entry the A register will contain the control code itself, while any parameters required will be stored in B (middle parameter if one exists) and C (last parameter).

```
                       ORG B533
FE0D       CONTROLS    CP "enter"
2884                   JR Z,ENTER              Jump to deal with "enter".
D610                   SUB 10
D8                     RET C                   Return with codes 00 to 0F.
FE06                   CP 06
2827                   JR Z,CTRL_AT            Jump to deal with AT.
D0                     RET NC                  Return with codes 17 to 1F.
C65F                   ADD A,CTRL_INFO lo
6F                     LD L,A
26B5                   LD H,CTRL_INFO hi       HL: points into control info table.
46                     LD B,(HL)               B:= bit mask for this control.
111400                 LD DE,0014              DE:= IX displacement to W_ATTR.
FE63                   CP 63
3802                   JR C,CTRL_CONT          Jump unless ctrl is INVERSE/OVER.
1E0B                   LD E,0B                 DE:= IX displacement to W_FLAGS.
DDE5       CTRL_CONT   PUSH IX
E1                     POP HL                  HL: pts to channel info block.
19                     ADD HL,DE               HL: points to variable to alter.
79                     LD A,C                  A:= control parameter.
48                     LD C,B                  C:= bit mask.
0F                     RRCA
07         CTRL_LOOP   RLCA
CB18                   RR B
30FB                   JR NC,CTRL_LOOP         A:= ctrl parameter, in correct posn.
AE                     XOR (HL)
A1                     AND C
AE                     XOR (HL)                Mix in reqd bits according to mask.
77                     LD (HL),A               Store variable.
C9                     RET

07         CTRL_INFO   DEFB 07                 Bit mask for INK.
38                     DEFB 38                 Bit mask for PAPER.
80                     DEFB 80                 Bit mask for FLASH.
40                     DEFB 40                 Bit mask for BRIGHT.
08                     DEFB 08                 Bit mask for INVERSE.
04                     DEFB 04                 Bit mask for OVER.
```

The next subroutine is the WINDOW version of the AT function. It performs the function AT B,C for the current window.

```
                       ORG B565
C5         CTRL_AT     PUSH BC
78                     LD A,B                  A:= proposed y coordinate.
CDB1B1                 CALL B1B1,LINE_A        Move print pos to start of this line.
C1                     POP BC
79                     LD A,C                  A:= proposed x coordinate.
A7                     AND A
C8                     RET Z                   Return if task already done.
DDBE0D                 CP (IX+W_WIDTH)
D29F1E                 JP NC,REPORT_B          Give error report if out of range.
DD770C                 LD (IX+W_XCOORD),A      Store new x coordinate.
0600                   LD B,00                 BC:= x coordinate.
DDCB0366               BIT 4,(IX+W_FLAGS)
281A                   JR Z,AT_EXIT            Jump with "Fast" channels.
E5                     PUSH HL                 Stack address of start of line.
DD4E17                 LD C,(IX+W_CH_WID)      BC:= character width in pixels.
60                     LD H,B
68                     LD L,B                  HL:= 0000.
09         AT_LOOP_1   ADD HL,BC
3D                     DEC A
20FC                   JR NZ,AT_LOOP_1         HL:= no of pixels to start of chr.
0603                   LD B,03
CB3C       AT_LOOP_2   SRL H
CB1D                   RR L
1F                     RRA
10F9                   DJNZ AT_LOOP_2          HL:= no of squares to start of chr.
07                     RLCA
07                     RLCA
07                     RLCA                    A:= pixel position within chr square.
DD7716                 LD (IX+W_PIX),A         Store in variable.
C1                     POP BC                  BC:= address of start of line.
09         AT_EXIT     ADD HL,BC               HL:= new print position address.
DD7510     STORE_ADDR  LD (IX+W_PRPOS)lo,L
DD7411                 LD (IX+W_PRPOS)hi,H     Store new print position.
C9                     RET                     Return.
```

The following subroutine is very short and very simple. It merely sets the attribute byte corresponding to the screen address in HL.

```
                       ORG B5A1
E5         SET_ATTR    PUSH HL
CDA5B1                 CALL B1A5,ATTR_ADDR     HL:= address of attribute byte.
DD7E14                 LD A,(IX+W_ATTR)        A:= current colours.
77                     LD (HL),A               Store attribute byte.
E1                     POP HL
C9                     RET                     Return.
```

This subroutine is intended for use with "Slow" windows only. It will plot one row of a character onto the screen.

```
                       ORG B5AB
CDA1B5     PLOT_ROW    CALL B5A1,SET_ATTR      Set attribute byte.
7E                     LD A,(HL)               A:= byte from screen.
AA                     XOR D
A0                     AND B
AA                     XOR D                   Mix in bits from character.
77                     LD (HL),A               Store in screen.
79                     LD A,C                  A:= low byte of mask.
3C                     INC A
280A                   JR Z,PLRW_EXIT          Exit if all bits stored on screen.
23                     INC HL                  HL: points to next screen byte.
CDA1B5                 CALL B5A1,SET_ATTR      Set this attribute byte as well.
7E                     LD A,(HL)               A:= byte from screen.
AB                     XOR E
A1                     AND C
AB                     XOR E                   Mix in bits from character.
77                     LD (HL),A               Store in screen.
2B                     DEC HL                  HL: points to original screen byte.
C38CB1                 JP B18C,DOWN_1          Point HL one pixel down, and return.
```

This subroutine will calculate 8* A+DE, and will also collect the current print position.

```
                       ORG B5C4
6F         PREPARE     LD L,A
2600                   LD H,00                 HL:= character code.
29                     ADD HL,HL
29                     ADD HL,HL
29                     ADD HL,HL               HL:= eight times character code.
19                     ADD HL,DE
EB                     EX DE,HL                DE:= address of pixel expansion.
DD6E10                 LD L,(IX+W_PRPOS)lo
DD6611                 LD H,(IX+W_PRPOS)hi     HL:= address of print position.
C9                     RET                     Return.
```

This subroutine will collect one byte from the pixel expansion pointed to by DE and will invert it if necessary.

```
                       ORG B5D3
1A         GET_ROW     LD A,(DE)               A:= next row of pixel expansion.
13                     INC DE                  DE: points to next row.
DDCB0B5E               BIT 3,(IX+W_FLAGS)
C8                     RET Z                   Return unless INVERSE 1 in operation.
2F                     CPL                     Otherwise invert the row.
C9                     RET                     Return.
```

This next and very important subroutine will actually print a character, specified in the A register, onto the current window.

```
                       ORG B5DC
F5         PRINT_CHR   PUSH AF                 Stack character to print.
F5                     PUSH AF                 Stack character to print (again).

DD7E0C                 LD A,(IX+W_XCOORD)      A:= current x coordinate.
DDBE0D                 CP (IX+W_WIDTH)
CCE8B2                 CALL Z,B2E8,ENTER       Print newline if at end of line.
DDCB0366               BIT 4,(IX+W_FLAGS)
2034                   JR NZ,PCHR_SLOW         Jump with "Slow" windows.
F1                     POP AF                  A:= character to print.
ED5B365C               LD DE,(CHARS)           DE:= addr of normal chr set -100h.
2012                   JR NZ,PCHR_OK_1         Jump with ASCII characters.
300A                   JR NC,PCHR_UDG          Jump with user defined graphics.
47                     LD B,A
CD380B                 CALL 0B38,PO_GR_1       Construct block graphic.
11925C                 LD DE,5C92,MEMBOT       DE: points to pixel expansion.
AF                     XOR A
1806                   JR PCHR_OK_1            Jump forward.
D690       PCHR_UDG    SUB 90
ED5B7B5C               LD DE,(UDG)             DE: points to user defined graphics.
CDC4B5     PCHR_OK_1   CALL B5C4,PREPARE       DE:= address of pixel expansion;
                                               HL:= address of print position.
CDA1B5                 CALL B5A1,SET_ATTR      Set attribute for this square.
E5                     PUSH HL
0608                   LD B,08                 B:= number of rows per line.
CDD3B5     PCHR_LOOP_1 CALL B5D3,GET_ROW       A:= next row from expansion.
CD71B1                 CALL B171,PAGE_7        Use RAM page 7 in case scrn 1 in use.
77                     LD (HL),A               Store row in screen.
24                     INC H                   HL: points to next row.
CD6CB1                 CALL B16C,PAGE_0        Restore RAM page zero.
10F3                   DJNZ PCHR_LOOP_1        Print whole character.
E1                     POP HL
23                     INC HL                  HL:= new print position.
C3B2B4                 JP B4B2,PCHR_EXIT       Jump to exit.
DDCB0356   PCHR_SLOW   BIT 2,(IX+W_FLAGS)
281E                   JR Z,PCHR_SLOW_2        Jump unless using double height.
DD7E0F                 LD A,(IX+W_HEIGHT)      A:= height of window.
3D                     DEC A                   A:= y coordinate of bottom line.
DDBE0E                 CP (IX+W_YCOORD)
2015                   JR NZ,PCHR_SLOW_2       Jump unless at bottom line.
DD4E0C                 LD C,(IX+W_XCOORD)      C:= current x coordinate.
47                     LD B,A                  B:= current y coordinate.
05                     DEC B                   B:= y coordinate after scroll.
DD7E0B                 LD A,(IX+W_FLAGS)
F5                     PUSH AF                 Stack flags.
C5                     PUSH BC                 Stack coordinates.
CDC3B2                 CALL B2C3,ENTER_1       Scroll the screen once.
C1                     POP BC                  BC:= coordinates.
CD65B5                 CALL B565,CTRL_AT       Move print position back where it
                                               belongs.
F1                     POP AF
DD770B                 LD (IX+W_FLAGS),A       Restore the flags.
F1         PCHR_SLOW_2 POP AF                  A:= character to print.
DD5E18                 LD E,(IX+W_CHARS)lo
DD5619                 LD D,(IX+W_CHARS)hi     DE:= addr of normal chr set -100h.
2008                   JR NZ,PCHR_OK_2         Jump with ASCII characters.
D680                   SUB 80
DD5E1A                 LD E,(IX+W_UDG)lo
DD561B                 LD D,(IX+W_UDG)hi       DE: points to graphics chr set.
CDC4B5     PCHR_OK_2   CALL B5C4,PREPARE       DE:= address of pixel expansion;
                                               HL:= address of print position.
E5                     PUSH HL                 Stack address of print position.
01FFFF                 LD BC,FFFF
DD7E17                 LD A (IX+W_CH_WID)      A:= width of chr in pixels.
CB38       PCHR_MASK_1 SRL B
CB19                   RR C
3D                     DEC A
20F9                   JR NZ,PCHR_MASK_1       BC:= mask, not yet in position.
DD7E16                 LD A,(IX+W_PIX)         A:= pix posn within chr square.
A7                     AND A
2808                   JR Z,PCHR_MASK_3        Jump if mask OK.
37         PCHR_MASK_2 SCF
CB18                   RR B
CB19                   RR C
3D                     DEC A
20F8                   JR NZ,PCHR_MASK_2       Otherwise rotate mask into place.
3E08       PCHR_MASK_3 LD A,08                 A:= number of rows per line.
F5         PCHR_LOOP_2 PUSH AF                 Stack loop counter.
CDD3B5                 CALL B5D3,GET_ROW       A:= next row from expansion.
D5                     PUSH DE                 Stack pointer into expansion.
57                     LD D,A
DD7E16                 LD A,(IX+W_PIX)         A:= pix posn within chr square.
A7                     AND A
2807                   JR Z,PCHR_ROW           Jump if pixels correctly alligned.
CB1A       PCHR_SHIFT  RR D
CB1B                   RR E
3D                     DEC A
20F9                   JR NZ,PCHR_SHIFT        Shift pixels into position.
CD71B1     PCHR_ROW    CALL B171,PAGE_7        Use RAM page 7 in case scrn 1 in use.
CDABB5                 CALL B5AB,PLOT_ROW      Plot row onto screen.
DDCB0356               BIT 2,(IX+W_FLAGS)
C4ABB5                 CALL NZ,B5AB,PLOT_ROW   And again if using double height.
CD6CB1                 CALL B16C,PAGE_0        Restore RAM page zero.
D1                     POP DE                  DE: points to pixel expansion.
F1                     POP AF                  A:= loop counter.
3D                     DEC A
20D8                   JR NZ,PCHR_LOOP_2       Print whole character.
E1                     POP HL                  HL:= original print position.
DD7E16                 LD A,(IX+W_PIX)         A:= original position within square.
DD8617                 ADD A,(IX+W_CH_WID)     Allow for width of character.
FE08       PCHR_POS    CP 08
3805                   JR C,PCHR_POS_2         Jump if print position OK.
D608                   SUB 08
23                     INC HL                  Otherwise amend it.
18F7                   JR PCHR_POS             Loop back to try again.
DD7716     PCHR_POS_2  LD (IX+W_PIX),A         Store new position within square.
CD9AB5     PCHR_EXIT   CALL B59A,STORE_ADDR    Store new print position.
DD340C                 INC (IX+W_XCOORD)       Increment x coordinate.
```

STREAMS AND CHANNELS

```
F1                      POP AF                   A:= character just printed.
C9                      RET                      Return.
```

The following subroutine will empty the buffer ("Slow" windows only), printing the former contents onto the window.

```
                        ORG B4BA
D4BBB2      EMPTY       CALL NC,B2BB,ENTER       Print a newline if required.
DDE5                    PUSH IX
E1                      POP HL                   HL: points to channel info block.
011D00                  LD BC,001D
09                      ADD HL,BC                HL: points to variable W_LEN.
7E                      LD A,(HL)                A:= number of characters in buffer.
A7                      AND A
C8          EMPTY_LOOP  RET Z                    Return if finished.
23                      INC HL                   HL: points to next chr in buffer.
7E                      LD A,(HL)
E5                      PUSH HL
CDF4B0                  CALL B0F4,CTYP_NOT_TOK   Flags indicate type of character.
CDDCB5                  CALL B5DC,PRINT_CHR      Print the character.
E1                      POP HL
DD351D                  DEC (IX+W_LEN)
18F0                    JR EMPTY_LOOP            Print all chrs in buffer.
```

The following subroutine will empty the buffer ("Slow") or do nothing ("Fast"). It will decide whether or not a newline is required, and print one if so.

```
                        ORG B4D6
DDCB0D66    EMPTY_2     BIT 4,(IX+W_FLAGS)
C8                      RET Z                    Return with "Fast" windows.
57                      LD D,A                   D:= next character to print.
DD7E1D                  LD A,(IX+W_LEN)          A:= length of word in buffer.
DD860C                  ADD A,(IX+W_XCOORD)      A:= potential x coordinate after
                                                 the buffer has been emptied.
DDBE0D                  CP (IX+W_WIDTH)
2002                    JR NZ,EMPTY_OUT          Jump unless word exactly fills line.
53                      LD D,E                   D:= alternative chr to print.
37                      SCF                      Signal "Newline not needed".
D5          EMPTY_OUT   PUSH DE
C5                      PUSH BC
CDBAB4                  CALL B4BA,EMPTY          Empty buffer with newline if needed.
C1                      POP BC
F1                      POP AF                   A:= next character to print.
C9                      RET                      Return.
```

And now at last we have the window output subroutine itself.

```
                        ORG B4F1
DD2A515C    WINDOW      LD IX,(CURCHL)           IX: points to channel info block.
CD87B0                  CALL B087,CHR_TYPE       Deal with keywords, etc.
F8                      RET M                    Return if tasks completed.
F5                      PUSH AF                  Stack character to print.
2824                    JR Z,WIND_ABLE           Jump with graphics characters.
3013                    JR NC,WIND_ASCII         Jump with ASCII characters.
A7                      AND A
CC92B2                  CALL Z,B292,CLS_WINDOW   Clear window for CHR$ 0.
F1                      POP AF
F5                      PUSH AF
5F                      LD E,A                   Signal "Chr not to be changed".
CDD6B4                  CALL B4D6,EMPTY_2        Empty buffer.
CD52B1                  CALL B152,CHR_TYPE_2A    Deal with TAB and comma control.
CD33B5      WIND_CTRL   CALL B533,CONTROLS       Deal with remaining ctrl characters.
F1                      POP AF
A7                      AND A                    Reset carry flag.
C9                      RET                      Return.
FE20        WIND_ASCII  CP "space"
200B                    JR NZ,WIND_ABLE          Jump with all chrs except "space".
1E0D                    LD E,"enter"             Use "enter" as alternative chr.
CDD6B4                  CALL B4D6,EMPTY_2        Empty buffer.
FE0D                    CP "enter"
28ED                    JR Z,WIND_CTRL           Jump if newline now required.
1806                    JR WIND_PRINT            jump to print "space" on window.
DDCB0D66    WIND_ABLE   BIT 4,(IX+W_FLAGS)
2004                    JR NZ,WIND_SLOW          Jump with "Slow" windows.
F1          WIND_PRINT  POP AF
C3DCB5                  JP B5DC,PRINT_CHR        Jump to print the character.
2A515C      WIND_SLOW   LD HL,(CURCHL)           HL: points to channel info block.
011D00                  LD BC,001D
09                      ADD HL,BC                HL: points to variable W_LEN.
7E                      LD A,(HL)                A:= number of chrs in buffer.
DDBE0D                  CP (IX+W_WIDTH)
2005                    JR NZ,WIND_SLOW_2        Jump unless buffer full.
E5                      PUSH HL
CDBAB4                  CALL B4BA,EMPTY          Empty buffer.
E1                      POP HL
F1          WIND_SLOW_2 POP AF
34                      INC (HL)                 Increment length of buffer.
4E                      LD C,(HL)
0600                    LD B,00                  BC:= new length of buffer.
09                      ADD HL,BC
77                      LD (HL),A                Store chr in buffer.
C9                      RET                      Return.
```

And finally – just a quick demonstration program to open a window and attach it to stream four. If you RUN this you'll be able to use PRINT #4; to print onto the window.

```
                        ORG B544
3E04        WIND_DEMO   LD A,04
08                      EX AF,AF'                A':= stream number.
2A7B5C                  LD HL,(UDG)              HL:= address of user defined graphics
0180FF                  LD BC,FF80
09                      ADD HL,BC
44                      LD B,H
4D                      LD C,L                   BC':= addr of UDG's minus 80h.
ED5B365C                LD DE,(CHARS)            DE':= addr of normal chr set -100h.
2607                    LD H,07                  H':= width of characters.
D9                      EXX
3E31                    LD A,31                  A:= attribute byte for window.
010102                  LD BC,0201               BC:= position of window.
111808                  LD DE,0818               DE:= size of window.
21FFFF                  LD HL,FFFF               HL: signals "Slow window" and
                                                 "Scroll pause enabled".
CD12B2                  CALL B212,OPEN_WINDOW    Open the window channel.
215827                  LD HL,2758
D9                      EXX                      HL':= value required by BASIC.
C9                      RET                      Return.
```

technical graphics

Toni Baker prepares you for a graphic encounter of the 3 dimensional kind.

In this article we start to look at the fundamentals of 3D — that is, three dimensional space. A solid object — a cabbage for instance, has three different dimensions — those of length, width and height. A flat object — such as a picture of a cabbage — has only two dimensions — length and width. As it happens, the image on the Spectrum's TV screen is flat (two dimensional). Any picture which appears on this screen must also be flat (two dimensional), which means that it is impossible to produce truly three dimensional images (ie solid images) on any TV, no matter how hard you try. This would require the use of a true three dimensional image system, such as a hologram. Some time in the future home computers may indeed be able to produce true 3D holographic images, but for the moment we are restricted to flatness. We can, however, create an *illusion* of depth. This process is known, perhaps mistakenly, as 3D-graphics.

The trick is to convert something which is three dimensional to a representation which is two dimensional. Take the cabbage for instance. Whilst a picture of a cabbage is only drawn onto a flat surface, it nonetheless *looks* like a real cabbage. This, then, is the key — we need a representation which is in fact a *picture* of something three dimensional. This task seems to be much easier for humans than it is for computers.

Things in space

The first thing we need to know is how to represent objects in three dimensional space. Imagine a piece of paper (or, if your imagination is not that good, use a real one). Now draw x and y axes on the paper, with the origin near the bottom left hand corner. The x axis goes off to the right, whilst the y axis goes up towards the top of the paper. Any point on the paper can be represented by two co-ordinates (x,y). This is two dimensional co-ordinate geometry. The PLOT command on the Spectrum uses this system, so you should be used to it.

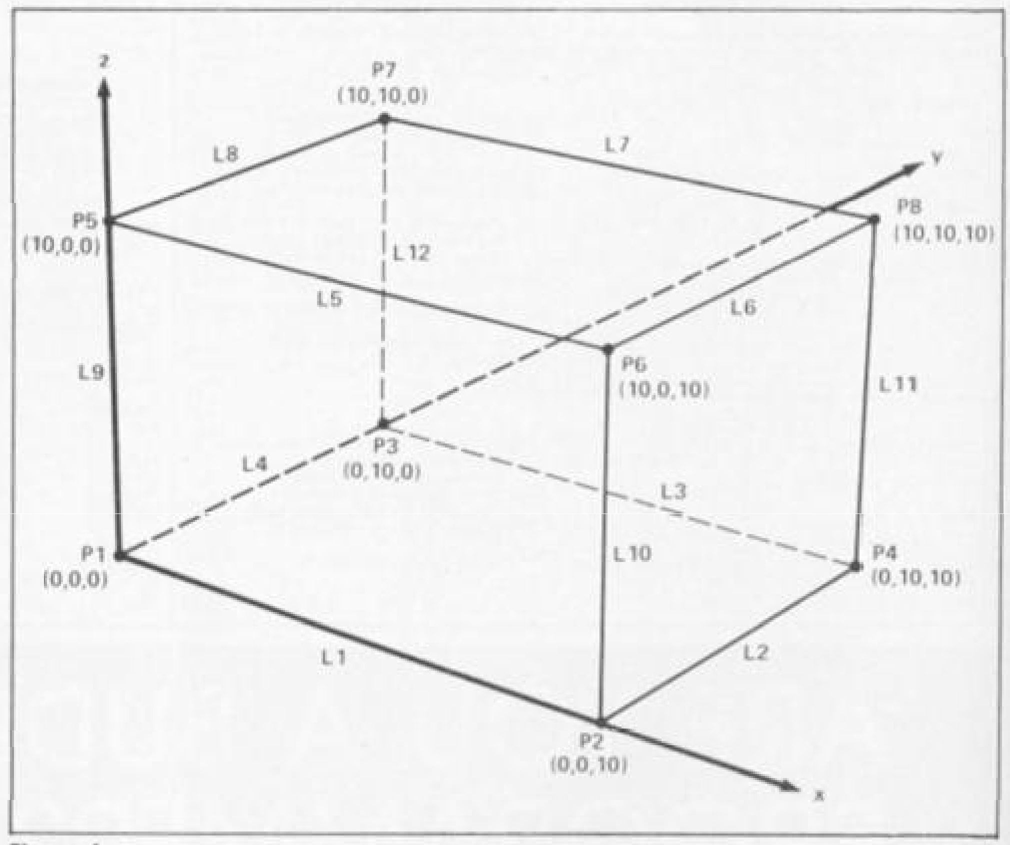


Figure 1

Now imagine a third axis, also emanating from the same origin. This axis is to go physically upwards, off the surface of the paper, away from the table on which the paper is resting, and up towards the ceiling of the room. This is the z axis. Any point in the room you are now sitting may be specified by three co-ordinates (x,y,z). For instance — take the paper itself. Any point on the paper which has two dimensional co-ordinates (x,y) also has three dimensional co-ordinates (x,y,0) with z being zero.

Imagine that a bumble bee enters the room and starts hovering just inches above the origin drawn on the piece of paper. Measure the height of the bumble bee above the origin, using the same units of measurement as the x and y axes are measured in, and preferably without being stung. Suppose the height of the bee was four units — the co-ordinates of the bee would be (0,0,4). Suppose it buzzed three units along in the direction of the x axis. Its co-ordinates would then be (3,0,4). Finally, suppose it flew two units in the direction of the y axis. Its co-ordinates would then be (3,2,4). This is three dimensional co-ordinate geometry.

Armed with this knowledge, we can now start to think about solid objects, and how they may be represented in this system. Imagine a cube, ten units along each side. Place the cube, in your mind, with one of its corners touching the origin. The cube should be sitting on the piece of paper with its edges running parallel to the x, y and z axes. Clearly, the co-ordinates of one of the corners is (0,0,0). It doesn't take too much imagination to figure out that the remaining seven corners have co-ordinates (0,0,10), (0,10,0), (0,10,10), (10,0,0), (10,0,10), (10,10,0) and (10,10,10), but these co-ordinates are not sufficient to define a cube — all they define are eight dots, four of them on the paper, and the other four floating in space ten units above the paper. What about the cube itself?

On edge

Since this series is concerned only with line drawings, the only thing we will need to know about the cube are its lines — or edges. This means that we need to record which points are connected to which other points.

Look at **figure one,** this is a drawing of a cube, but with every vertex (corner) marked with the letter P and a number between one and eight, and with every line marked with the letter L and a number between one and twelve. Notice that there are more lines than there are vertices.

Figure two shows a BASIC program which draws the cube in figure one. Don't worry too much at the USR statement in line 540 — it's just the points and lines we're interested in at the moment. Line 10 dimensions an array P() to hold all the points. It is dimensioned eight by three because there are eight points, and three co-ordinates for each point. Line 20 dimensions an array L() to hold all the lines. It is dimensioned twelve by two because there are twelve lines, and two points at the two ends of each line.

Lines 30 to 110 initialise the array P() to hold the co-ordinates of each of the corners of the cube in such a way as to maintain the numbering in figure one. Lines 120 to 190 initialise the array L(). Note that this time we have to use data because there's no easy mathematical way to work them all out as there was with the vertices. The rest of the program just draws the cube. You can run this program if you like, but make sure that the machine code is in place first.

Incidentally, if you change lines 60 to 80 so that they end 8*K, 10*J and 12*I respectively then you'll get a *cuboid*, not a cube — a rectangular block. Try it — it presents a much more pleasing picture because the front and back corners don't overlap.

This brings us to the most important question of all. How does it all work? We need to understand the general principle of converting a three dimensional solid object down to a two dimensional picture. Look agains at figure one. Notice that, for instance, line L7 is connected to points P7 and P8 — but figure one is a picture, not a real cube. In other words, line L7 is connected to points P7 and P8 both in the real three dimensional cube, and in the two dimensional picture. This is true for all of the lines, not just for L7. Although this may seem stunningly obvious, it is nonetheless the most important piece of information in 3D graphics. It means that if you can work out whereabouts on the screen the image of P7 will fall, and if you can also work out whereabouts on the screen the image of P8 will fall, then it is obvious that the image of the line L7 will just be a straight line connecting the image of P7 with the image of P8. This we can do on the Spectrum's screen using PLOT and DRAW as normal in BASIC. All we now need is a method for working out the position on screen of the images of all the points.

There are many, many methods of transforming three dimensional co-ordinates down to two dimensional co-ordinates. The simplest possible means is just to throw away the z co-ordinate leaving just x and y. This gives you a plan view of the object — not very satisfactory, however — we need something a bit more daring than that.

Projection

The method we shall use is a technique called Isometric Projection. The idea is that you have to imagine a camera floating in space looking at the object. In isometric projection the camera is always located at co-ordinates (N,N,N). N can be any, very large, positive number — the larger the better, since the camera is assumed to be a long way from the origin. The camera is pointing directly towards the origin. It is the right way up, and it has a very powerful zoom lens, so it can see the object (which is located at or near the origin). The image that the camera would see is the picture which is to appear on the screen.

There are other types of projection (*many* other types), which have the camera and the object at different positions in space, but the idea is always the same — what the camera sees, the Spectrum draws.

In future issues, I will show you how to use all these other projections, but for now we shall concentrate on isometric. It is sufficiently powerful to be able to demonstrate the basic ideas of 3D and projection, whilst at the same time it is sufficiently simple (mathematically speaking) so that anyone who knows anything about BASIC will be able to understand it.

Let's look at the mathematical side of things first, shall we? Suppose a point in three dimensional space has co-ordinates (x,y,z) — any point will do. Suppose also that the image of this point appears on the screen with PLOT co-ordinates (p,q). What we need to know is how we can calculate p and q, given only x, y and z.

The solution turns out to be so easy that we can do the task in BASIC. The following two LET statements will make the projection:

```
10 DIM P(8,3)
20 DIM L(12,2)
30 FOR I = 0 TO 1
40 FOR J = 0 TO 1
50 FOR K = 0 TO 1
60 LET P(4*I+2*J+K+1,1) = 10*K
70 LET P(4*I+2*J+K+1,2) = 10*J
80 LET P(4*I+2*J+K+1,3) = 10*I
90 NEXT K
100 NEXT J
110 NEXT I
120 FOR I = 1 TO 12
130 FOR J = 1 TO 2
140 READ L(I,J)
150 NEXT J
160 NEXT I
170 DATA 1,2,2,4,4,3,3,1
180 DATA 5,6,6,8,8,7,7,5
190 DATA 1,5,2,6,4,8,3,7
200 FOR I = 1 TO 12
210 LET A = 1: GO SUB 500
220 LET P1 = 5*P+128
230 LET Q1 = 5*Q+88
240 PLOT P1,Q1
250 LET A = 2: GO SUB 500
260 DRAW 5*P+128-P1,5*Q+88-Q1
270 NEXT I
280 STOP
500 LET A = L(I,A)
510 LET X = P(A,1)
520 LET Y = P(A,2)
530 LET Z = P(A,3)
540 RANDOMIZE USR 33320
550 RETURN
```

Figure 2

LET p=SQR(3) * (y—x) / 2
LET q=z — (y+x)/2

In other words, we can do the whole task in BASIC — we don't need any machine code at all. We can define the points in space and the lines joining them; we can transform the points using the above formulae; we can PLOT the new points, and we can DRAW the connecting lines. All very easy.

The machine code I have included is really only intended to work out the above formulae — that is — given x, y and z the machine code will work out the values of p and q. Despite being in machine code, the machine code program uses the values from BASIC variables, and assigns BASIC variables with the results. You may care to examine the machine code to see how this is achieved. None of it is really difficult — it all hinges on the way that BASIC variables are ▶▶

Listing 1

```
                        ORG 8200
2A4B5C     SEARCH_VAR   LD HL,(VARS)             HL: points to variables area.
7E         S_V_LOOP     LD A,(HL)                A:= next variable byte.
E67F                    AND 7F                   Ignore bit 7.
37                      SCF
C8                      RET Z                    Return with carry set if byte 80h
                                                 reached (ie if variable not found).
B9                      CP C
C8                      RET Z                    Return with carry reset if
                                                 variable found.
C5                      PUSH BC                  Stack variable name searched for.
CDB819                  CALL 19B8,NEXT_ONE       DE: points to next variable.
EB                      EX DE,HL                 HL: points to next variable.
C1                      POP BC                   C:= variable name.
18F1                    JR S_V_LOOP              Jump back to continue search.

                        ORG 8212
0E7A       STK_ZYX      LD C,7A                  C:= code for variable Z.
CD1E82                  CALL 821E,GET_VAR        Stack variable Z.
0E79       STK_YX       LD C,79                  C:= code for variable Y.
CD1E82                  CALL 821E,GET_VAR        Stack variable Y.
0E78                    LD C,78                  C:= code for variable X.
CD0082     GET_VAR      CALL 8200,SEARCH_VAR     Search for variable.
DA2E1C                  JP C,1C2E,REPORT_2       Error if variable not found.
23                      INC HL                   HL: points to variable contents.
C3B433                  JP 33B4,STACK_NUM        Stack the variable contents onto
                                                 the calculator stack, and return.

                        ORG 8228
CD1782     TRANSFORM    CALL 8217,STK_YX         Get Y and X onto calculator stack.
EF                      RST 28                   Y,X
03                      subtract                 Y-X
3440B00003              stk data 3               Y-X,3
28                      sqr                      Y-X,SQR(3)
04                      multiply                 SQR(3)*(Y-X)
A2                      const half               SQR(3)*(Y-X),1/2
04                      multiply                 SQR(3)*(Y-X)/2
38                      endcalc
0E70                    LD C,70                  C:= code for variable P.
CD4782                  CALL 8247,ASSIGN_VAR     LET P = SQR(3)*(Y-X)/2.
CD1282                  CALL 8212,STK_ZYX        Get Z,Y,X onto calculator stack.
EF                      RST 28                   Z,Y,X
0F                      add                      Z,Y+X
A2                      const half               Z,Y+X,1/2
04                      multiply                 Z,(Y+X)/2
03                      subtract                 Z-(Y+X)/2
38                      endcalc
0E71                    LD C,71                  C:= code for variable Q.
CD0082     ASSIGN_VAR   CALL 8200,SEARCH_VAR     Search for variable.
300A                    JR NC,ASSIGN_VAR_2       Jump if variable located.
C5                      PUSH BC                  Stack code for variable name.
010600                  LD BC,0006
CD5516                  CALL 1655,MAKE_ROOM      Create room for variable.
23                      INC HL                   HL: points to start of new room.
C1                      POP BC                   C:= code for variable name.
71                      LD (HL),C                Store the variable name.
23         ASSIGN_VAR_2 INC HL                   HL: points to variable contents.
E5                      PUSH HL                  Stack address of variable contents.
CDBF35                  CALL 35BF,STK_PNTRS      HL:= address of last item on stack.
22655C                  LD (STKEND),HL           Delete the item from the stack.
D1                      POP DE                   DE: points to variable contents.
010500                  LD BC,0005
EDB0                    LDIR                     Copy number into variable.
C9                      RET                      Return.
```

stored in the variables area.

In next month's article — which will be the final part of this series — we shall look at isometric projection a little deeper. We shall use it to draw three dimensional graphs, using the simplest of hidden-line algorithms so that lines which would be hidden to the camera won't be drawn on the screen. And — oh yes — there will be *lots* of machine code involved.

Ray Elder on adding additional commands.

We have received several enquiries from avid 81ers about the possibility of adding RESTORE, DATA and READ to the ZX81.

There have been several methods which have simulated these operations such as storing DATA in REM lines and PEEKing it into variables, but the most efficient that I have encountered was written by our own regular writer David Nowotnik for us in the heyday of the ZX81 in 1983. As many existing users will undoubtedly have missed this system I have no hesitation in reprinting it, especially in light of the recent requests for such a routine.

Restore/Data/Read

The program consists of two routines which must be located in the first REM line statement of the program. The length is a mere 128 bytes, but the line 1 REM . . . must contain 132 dots (or characters of your own choice).

Enter program 1 and SAVE it in case of any errors, then RUN it and delete every line EXCEPT line 1. This is done by entering each line number one by one.

Now enter the lines in program 2, these are needed for every program that you wish to use the functions in, the rest of your programs being written after these lines. Once more SAVE the whole thing to tape. This is your master copy.

Program 3 is a demo of how you may use the routines.

RESTORE resets the program pointer to the start of the DATA line and is used by the command RAND USR 16520.
READ has two possible forms, **LET C$=A—$(TO RAND USR 16530)** to read string or character data, and **LET C=VAL A$(TO RAND USR 16530).**
DATA lines are stored in any line but with REM and the graphic obtained on shifted key A following it (a 'grey' square). NOTE that in program 3 this does not show up on our printer so be sure to add it in.

9999

A final indicator to the routine that it has reached the end of the program is required and this is provided in the form of a line 9999 REM followed by the inverse space (black square). This line is ESSENTIAL.

The DIM A$ and CLEAR commands instructions are important in the main program as it ensures that the variable A$ is the first in the variable area of memory and the main routine can locate it.

You may find that LIST produces just 1 REM, to overcome this use LIST 2 (or 3 or 100 or 3000 etc.).

Data REMs should have the number of data items as the first character following the grey square after the REM. For example:

170 REM 4,EENY,MEENY,MINY,MO

Finally, note that you cannot mix numeric and string data on the same data line, two or more lines will be required and dummy READs will be needed to jump over the unrequired data.

RAM Packs

We received a letter from Philip C. Allen who asks for details of 64K RAM packs, joystick interfaces, and fast storage (disks etc.) available for the ZX81.

As far as we can tell there are not longer any companies producing ZX81 equipment, and not any likelyhood of any starting to either. If anyone has such devices or knows where they may be obtained then we would be only too pleased to pass on such info.

Philip also asks for details of ZX81 user groups and he should have a copy of last month's page by now which mentions a few such groups. We will continue to publish any information anyone cares to supply us with for the benefit of you all. Philip himself sent us some details of a company in Birmingham, "House of Software", 51 Snowhill Queensway, which has some stocks of ZX81 software available. So if that's your area amble along and check them out for yourself.

Bye all . . . . . .

```
   1 REM.......................
...............................
...............................
...............................
....
   2 FOR I=16514 TO 165651
   3 LET A$="7676ØØØØØØØØ2184407
523742336ØØC92A844Ø3A864ØA72Ø187
EFEEA28Ø32318F8237EFE8Ø283CFEØ82
ØEF233EØ132864ØEB2A1Ø4ØØ1Ø6ØØØ9E
B3EØØ32874Ø7EFE1A2813FE7628ØAEDA
ØE521874Ø34E118ED3EØØ32864Ø23228
44ØØ6ØØ21874Ø4EC9ØØ2AØC4ØØ1F8Ø2Ø
9EB21FB4ØØ111ØØEDBØCD884Ø1898ØØ3
43A39ØØ342BØØ29263926ØØ2A3737343
7"
   4 POKE I, 16*(CODE A$-28)+COD
E A$(2)-28
   5 LET A$=A$(3 TO )
   6 NEXT I
```

Program 1

```
   1 REM.......................
....CONTAINING THE CODE FROM....
...............................
....PROGRAM 1..................
....
   9 REM 1653Ø=READ, 1652Ø=REST
  1Ø CLEAR
  2Ø DIM A$(32)
  3Ø RAND USR 1652Ø
9999 REM
```

Program 2

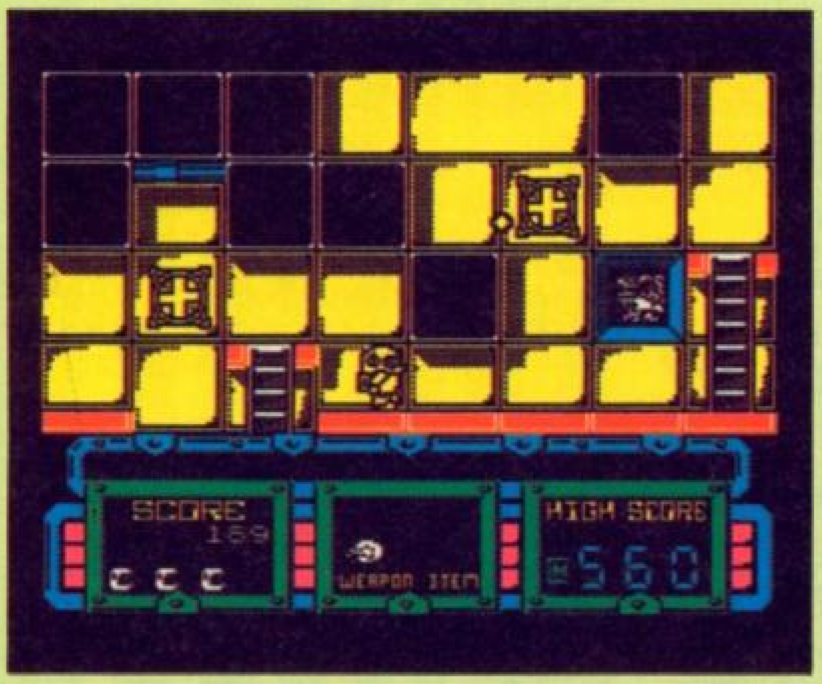
```
   1 REM.......................
....CONTAINING THE CODE FROM....
...............................
....PROGRAM 1..................
....
   9 REM 1653Ø=READ, 1652Ø=REST
 1ØØ LET C=VAL A$( TO RAND USR 1
653Ø)
 11Ø DIM Z$(C,1Ø)
 12Ø FOR I=1 TO C
 13Ø LET Z$(I)=A$( TO RAND USR 1
653Ø)
 15Ø PRINT Z$(I)
 16Ø NEXT I
 17Ø REM  4,EENY,MEENY,MINY,MO
```

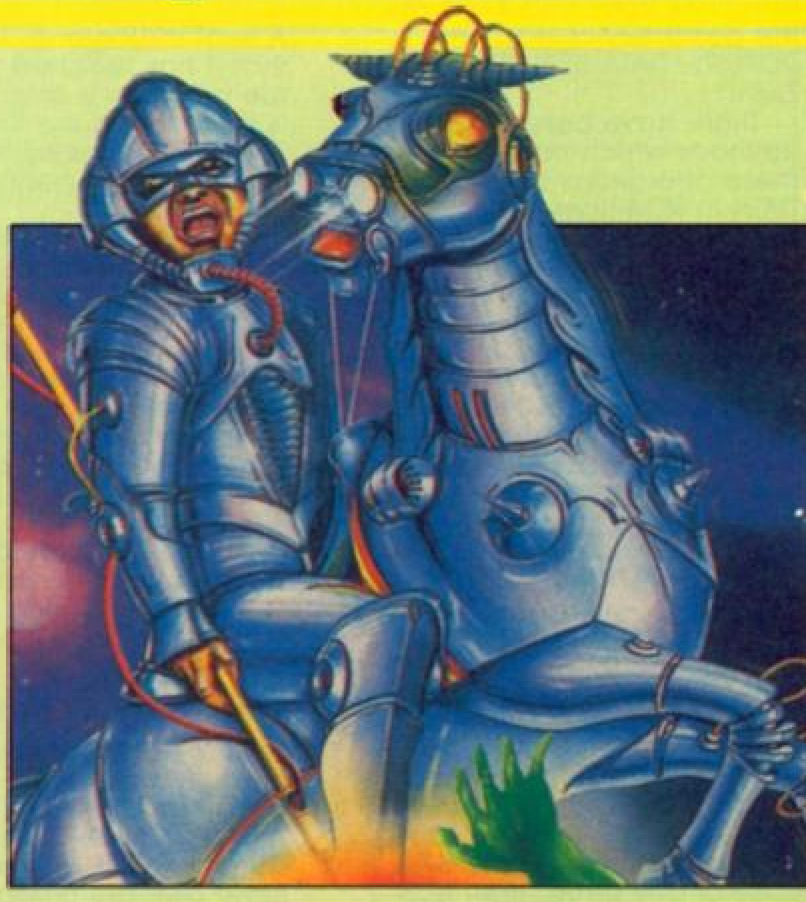
Program 3

PAGE 81

FUTURE KNIGHT

Future Knight proves hard to define but easy to enjoy





Future Knight
Gremlin
£7.95

Spectrum gamers that enjoy categorising games will have fun with this latest offering from Gremlin.

Is it a shoot em up, or a platform game or perhaps an arcade adventure? Or is it a completely new style of game combining the action of all three? Either way it's going to be big.

The plot revolves around our attempts as Randolph the hero to rescue your beloved maiden from the evil clutches of the evil Spegbott the Terrible.

Wearing your Omnibot Mark IV all purpose attack suit, complete with laser assisted rifle you rush to answer an inter-dimensional distress signal and arrive in the S.S. Rustbucket. However, instead of finding your Princess Amelia you're greeted by berserk defence droids that swamp you and drain your life energy.

Luckily, you've brought a couple of spare lives with you in case you lose all 999 of your energy points.

These defence droids come in many shapes and sizes and range from high flying ghosts to slithering blobs of goo.



Bubbling lava

To add to your problems there's also deadly pools of bubbling lava and platform traps that you can leap into but can't jump, walk or blast your way out.

The game begins inside the crashed Rustbucket and your first job is to find the way out onto the planet then search a jungle until you find Spegbott's castle and eventually your Princess. Ahead of you lies 20 levels of



vertically scrolling screens that form the maze of ladders, platforms and hazards of the Rustbucket and the planet outside.

All is not lost as help is at hand in the objects that you can find around the ship, although you will have to fight for them.

Safe passes and securo keys open and unlock the exit doors that lead from one level to another until eventually you find the exit pass to let you out of the ship. You may also find bombs that destroy a screenful of critters while replenishing your energy as well as Confusers to stun them and the mysterious Shorteners and Flash Bangers.

Henchodroid

Once you reach the castle and find the dungeon you will have to defeat the almost indestructable Henchodroid. You'll probably achieve this through objects that you've found but first you will have to perform a cosmic juggling act as you can only carry one object at a time!

Despite this restriction you'll soon be bounding through the levels.

Unfortunately, you'll have to do the full 20 levels in one sitting as there isn't a save option or even a pause button. Leave the game for a few minutes and Randolph will wave to attract your attention and then spin around losing energy at an alarming rate.

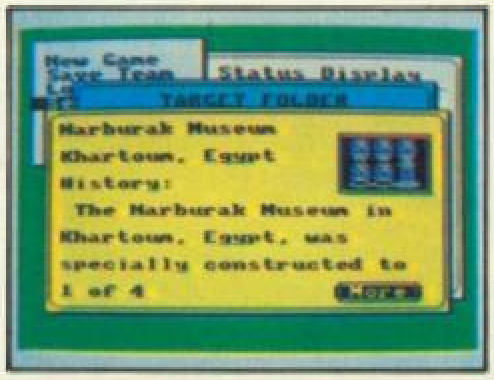
A superb mixture of all that's best in arcade adventures, platform games and shoot em ups combined to signal monster hit.



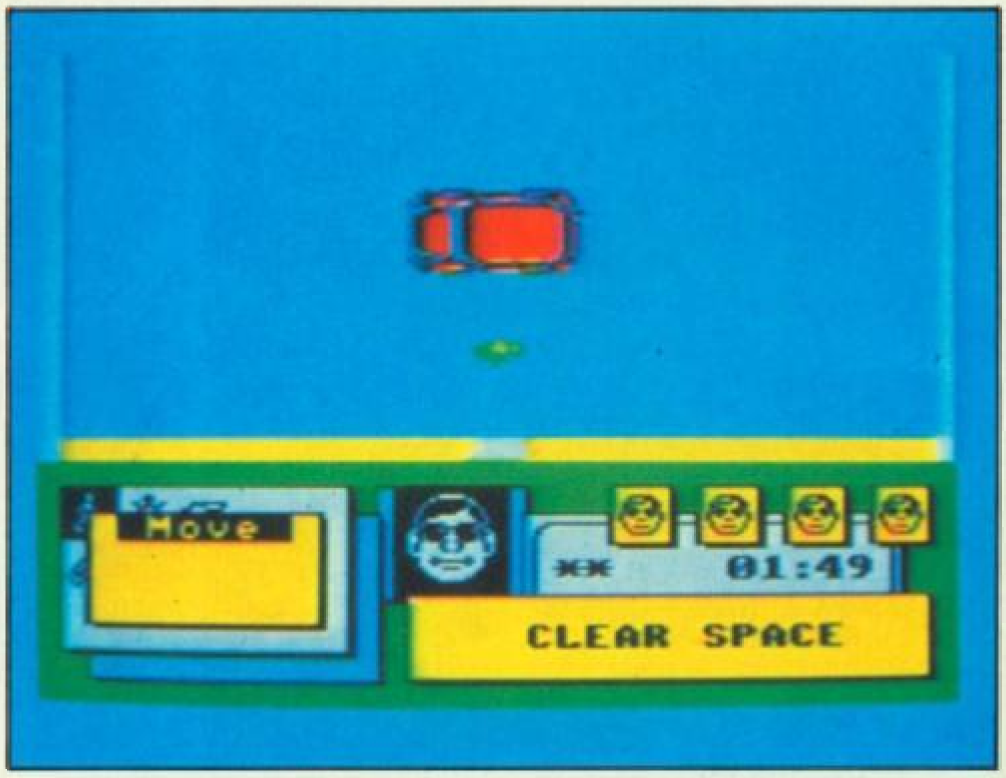
MONSTER HIT

FUTURE KNIGHT

They stole a million



A big time crime simulation from Ariolasoft.



They Stole a Million
Ariolasoft
£8.95

You are the Boss of a gang that's decided to hit the big time. Gone are the daring days of riding on the buses without a ticket. Ahead of you lies a life of crime helped by your S.W.A.G. (Software for Aspiring Gansters) disk.

Through S.W.A.G. you can select your target from Coin Dealers to Banks and buy information and blueprints so that the team you hire is right for the job.

Each team member has their own special skills from safe blowing by Detonate D'Arcy to electronics expert Charlie Volt.

Once you've chosen your team and found the right fence you can plan the job.

In this phase you plot the exact movements and actions of each team member throughout the raid using the joystick or keyboard controlled icons. Therefore you can make sure that Skeleton Joe has picked all of the locks to let D'Arcy through to get the swag. Get the timing wrong and you could end up serving time!

Then it's for the raid itself. If your plan works well you and your team will soon be richer. But the best laid plans . . .

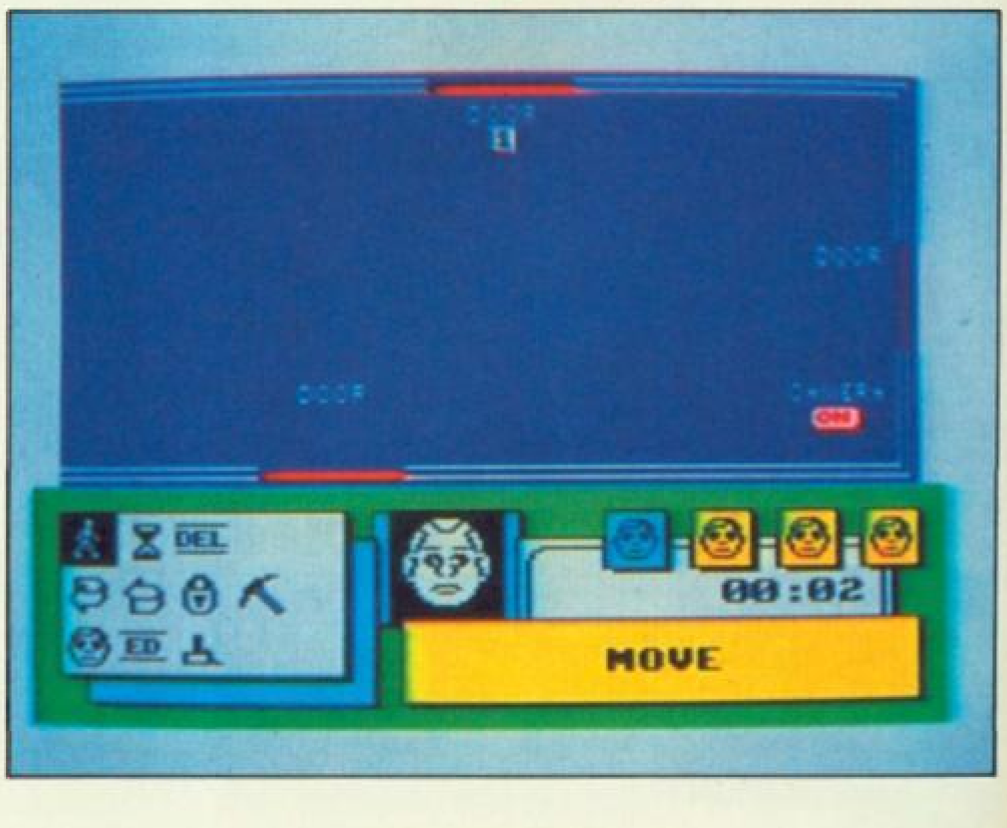
If any robber has a problem then he'll radio for assistance then you can either give him extra instructions or go and sort him out.

It's important to get your team right as the wrong person doing the wrong job can land you in prison. Even the lookout is important as nosey police cars must be spotted and the robbery halted until the lookout gives the all clear.

If you succeed you and your team will be richer and able to plan bigger and better jobs until finally you get the chance to steal a million.

If you're prepared to do the planning and research you'll find this game fascinating but fairly soon hit all the targets and finish the game.





UPGRADE ART

Both The Artist and Art Studio are re-released this month in enhanced versions, but are the improvements worth having?

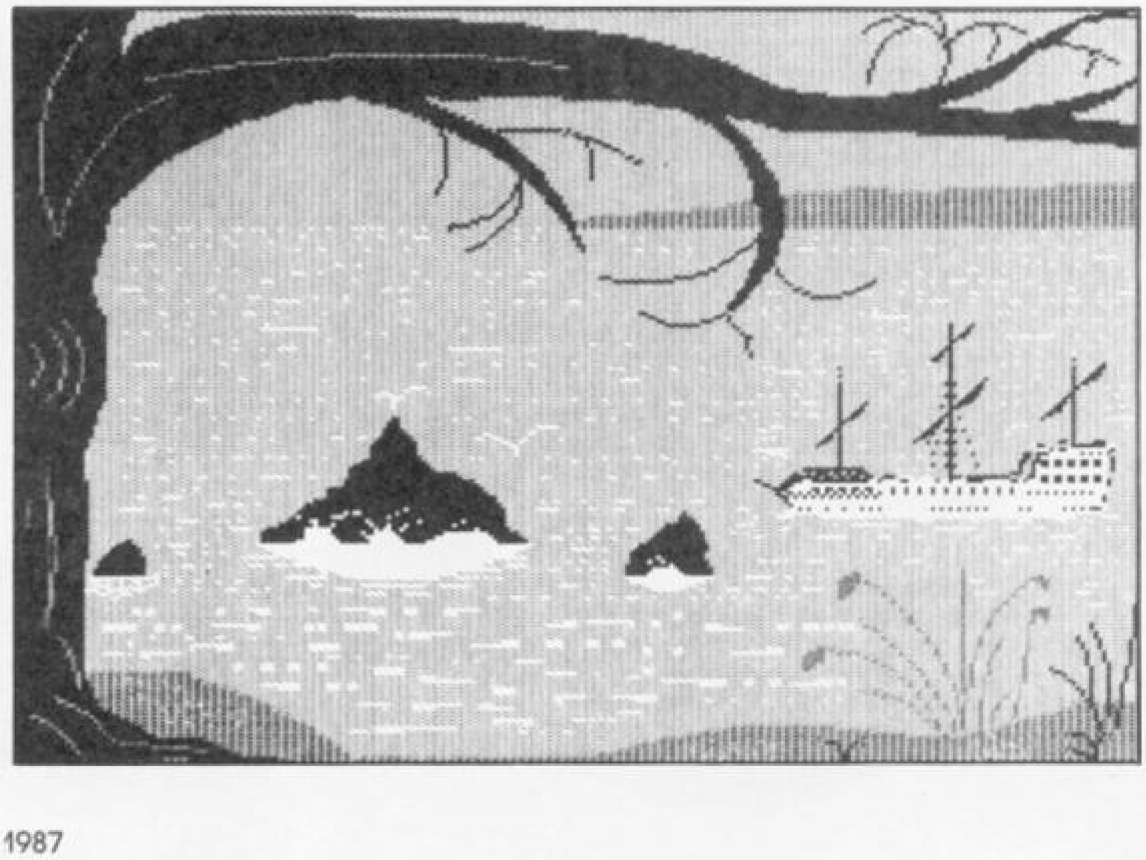
The Artist II
Softechnics
£14.95

I have mixed feelings about this program. It could — and should have been by far the best package of its type available for the Spectrum, but it seems to have been rushed on to the market without enough checking, and a handful of bugs have been allowed to take the edge off it. Some of them are just irritating things which don't matter too much — the storage menus are the wrong way round, so you have to select 'tape' to use Microdrive and vice versa; if you move the screen up to work on the part normally hidden by the icon menus, you have to scroll the screen after some options, because the bottom three lines are transferred to the top when you return to the normal viewing screen. But some other problems are much more serious. The SAVE/LOAD operations do not work when you are using the design font option. SAVE stores the wrong block of memory, and LOAD crashes the program! I have managed to find a way of getting round this (see footnote), but the deficiencies in the printing facilities have defeated me so far.

The handbook says that the program will drive an Opus disc drive centronics interface or Kempston E. I am told that there are no problems with Opus, but with my Kempston E, only the grey scale screen dumps would work. The ordinary screen dumps and the Pagemaker printing option simply produced the required number of line feeds but no printing. It is especially galling to use the Pagemaker, an exciting facility which allows you to produce an illustrated A4 page, combining text produced by Softechnics' word processor, The Writer, with graphics produced by The Artist II. There you are, with your beautiful illustrated page on screen, and because the print option doesn't work, you can't get it on to paper.

And yet . . .

But, despite the bugs, this is still a very powerful package. Developed from Softechnics' earlier success, The Artist, it now supports Microdrive, Opus disc or tape storage, keyboard or Kempston joystick, Kempston or AMX mice for control. The layout has been completely redesigned, and now has easy to use pull-down and icon menus. I am sorry to see that the facility to draw an arc between two points is no longer with us, and that the keyboard cursor control keys are still letter keys rather than the arrows, but the extra facilities the program now has are tremendous. There are now elastic lines, circles, ellipses or rectangles, and shapes can be drawn in outline or ready-filled with the chosen texture. There are 28 textures available, and all are redefinable and can be saved or loaded. The fill option is as efficient as ever, and the enlarge option, for detailed work, far better than in the old program. The enlarged window is shown alongside the same area in normal size so that the



UPGRADE ART

effect of changes can be seen as you work.

For lettering, the program comes with five fonts. These are redefinable, but you would be well advised to confine your modifications to fonts 3, 4 and 5. Font 1 is the Spectrum character set, and being held in ROM, ignores all your attempts to modify it, though the font designer gives the impression that you are making changes. Font 2, the small typeface, is used extensively by the program. I discovered the hard way — the handbook does not warn you — that inverting it makes the menu cursors invisible and mirroring it makes the menus unreadable. It was virtually impossible to get back to normal without reloading the program.

Cut 'n paste

The window and the cut and paste options are the program's great strength. A rectangular window of any size (corresponding to the character squares) can be defined anywhere on screen and the area within it cleared, moved, enlarged or compressed, rotated, inverted or mirrored. The design can be thickened or outlined, attributes changed, or the image scrolled. Cut and paste has some of these facilities, but any size or shape of area can be manipulated and, whereas the window option only allows portions to be moved in character-square jumps, the scroll option in cut and paste allows placement to pixel accuracy. Both window and cut and paste have an insert mode which allows a second screen to be loaded, and portions of it cut and inserted into the current artwork. The ship in the illustration was cut from one of the demo screens supplied with the earlier Artist program, and inserted into the seascape drawn with this one. This is a very powerful facility, allowing you to build up a screen library and bring bits and pieces of several screens together in a new one.

There is now a separate sprite designer, which has normal sized and enlarged screens upon which sprites up to 6 x 6 character squares can be designed. They can also be 'grabbed' from existing screens and inserted into the present one. Sprites for animation can be designed and stored in a sequential file — up to 79 screens 3 x 3 square size, less for larger. To test animation, the speed and frame numbers are selected and the animation is demonstrated on the normal size screen. Sprite files can be saved with their frame information — for reloading into the designer — or as a string of bytes for use in other programs.

Is The Artist II a good buy? Well . . . if you have an Opus disc drive, or are not particularly interested in screen dumps, yes. You will not find a better or more powerful screen art program than this one. But if, like me, you use screen dumps a lot, you might find it disappointing. I would like to think that Softechnics will do some more work on it and issue a Mark 2 version without the bugs, driving the interfaces it is supposed to drive, and with a better handbook than it now has. The present one is rather sketchy — you almost have to read between the lines to discover the full potential of some of the program's options — and it has too many printing errors. Dare I also suggest that a free tape exchange for those who bought this flawed version would be a nice gesture? But certainly, Softechnics should take another look at it. it is far too good a program to be left in the state it is now. *Carol Brooksbank*

This is a footnote

To save and load type fonts.

Select the save/load option and give the file name when prompted. Use the BREAK key to return to BASIC. (Do not start the tape if saving). Enter as a direct command;

LET B=number

number=62268 for font 3
61500 for font 4
63036 for font 5

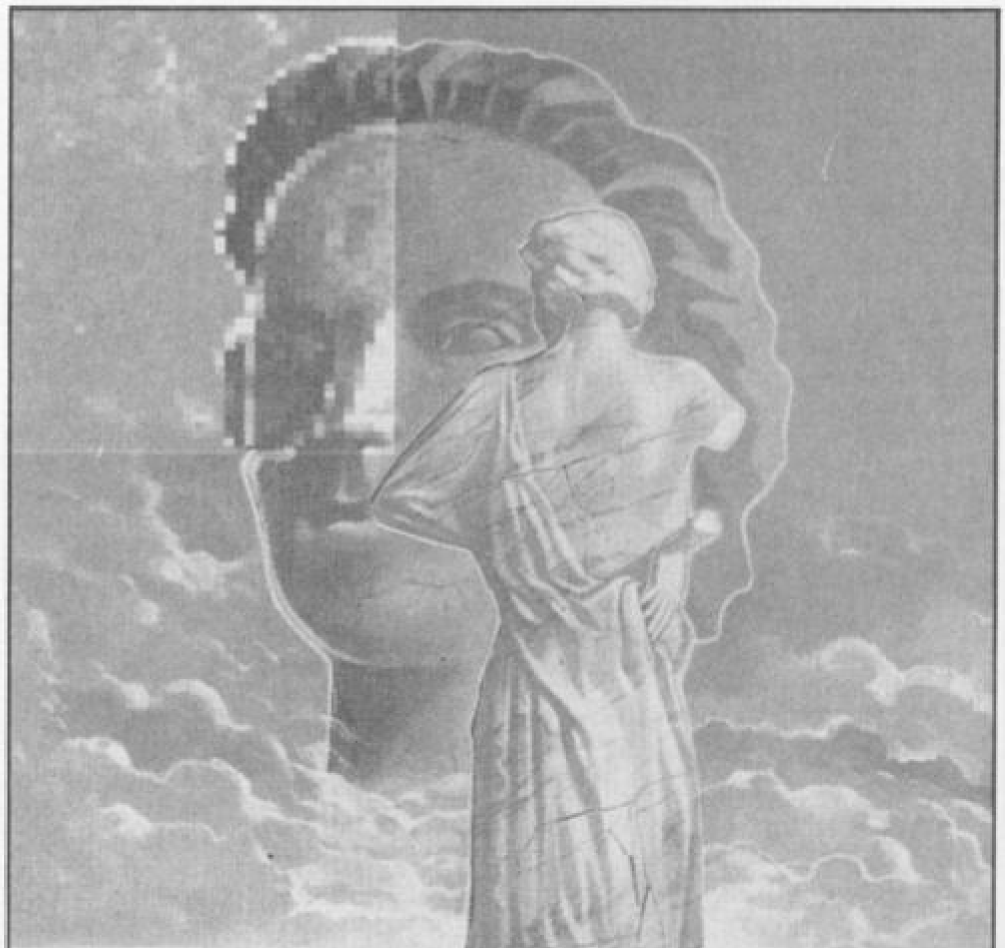
Now enter GO TO

72 to load from microdrive/disc
74 to load from tape
82 to save to microdrive/disc
84 to save to tape

Proceed as usual to save/load

Advanced OCP Art Studio
Rainbird
£24.95

It arrived too late for a Christmas review, but the new Advanced Art Studio should have gotten into the shops in time to make a nice little prezzie for anyone who was lucky



enough to get a 128 from Santa.

This enhanced version of OCP's Art Studio (which is specifically for the 128 and won't run on any of the 48K versions of the Spectrum) uses the 128's additional memory mainly for storage purposes, giving you a 42K RAM Disc facility as well as a 16K 'Scrapbook' which, between them, allows you to store a number of screens, character sets, Fill patterns and so on, and to call them back from memory instantly — so saving you all the fuss of Saving and Loading to and from tape all the time.

RAM what?

In case you're not familiar with that bit of jargon, a RAM Disc is an area of memory that is set aside purely for storage of programs, data, or, in this case, screen pictures and patterns. Anything stored in this area simply sits there until you need it and can then be summoned up instantly with just the press of a button.

The new storage facilities are implemented by adding a new sub-menu to some of the existing command menus and treating the RAM Disc almost as if it were a microdrive. Suppose that you're halfway through designing the loading screen for the latest mega-game when you decide that you want to call up a new character set for printing the name of the game. You push the cursor over to the 'File' window as you normally would, but when the menu appears asking whether you want to save your picture to tape or microdrive you choose the microdrive option. This leads you to a new sub-menu which allows all the usual options for dealing with microdrives, but also has a new option for the RAM Disc, as well as a catalogue listing all the files on RAM Disc or microdrive.

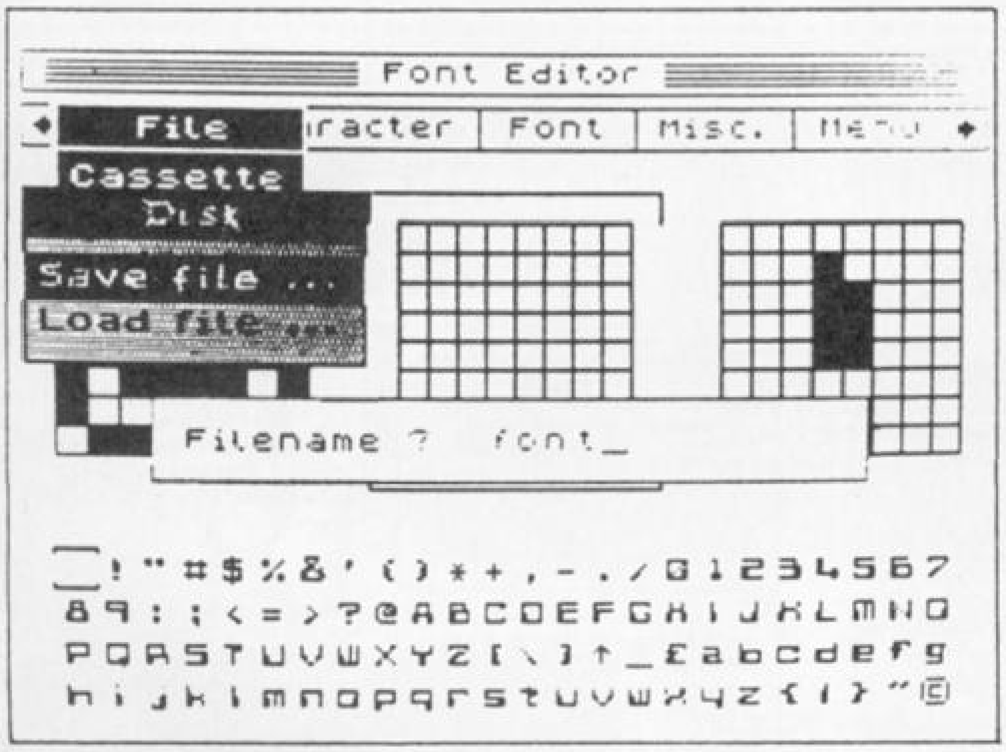
So, you simply give your picture a name and instantly save it onto the RAM Disc. The saved picture is automatically verified at the same time, eliminating the business of saving and verifying onto tape. Next, you choose the 'Text' menu and select the new command, 'File Menu', which leads once more to the cassette/microdrive choice. A quick look at the catalogue shows you that Rainbird have thoughtfully included a few alternative character sets which are tucked neatly away on the RAM Disc. You select whichever set you think is suitable (if you want to create a typeface of your own there's a 'blank' character set which can be edited, allowing you to do this), call back your picture and get back to work. This whole process takes just a few seconds whereas it could take minutes if you were relying purely on tape storage.

As well as these alternative character sets Rainbird have also included a couple of sets of 'Brush' and 'Fill' patterns to give you a bit of extra variety, or you can always create your own and file them away on RAM Disc. With over 40K of RAM Disc to play with there's plenty of room for all sorts of bits and pieces, and if you use that up you can always save the RAM onto tape and start on a new 'disc'.

The Scrapbook facility is a sort of souped up 'cut and paste' option in that it gives you 16K worth of memory to store small sections of larger pictures, so that you can use these same sections over and over, quickly transferring them from one picture to another.

As far as the business of drawing pictures is concerned there's hardly anything that could be created with the enhanced Art Studio that couldn't be created on the original 48K version (although the 128 version does include a new 'arc' command), however the new fast storage offered by this version is almost as good as fitting your Spectrum with a disc drive.

I suppose it's a tribute to the quality of the original program that it can't be much improved upon even with an additional 80K to play with, and owners of mere 48K machines aren't going to be left too far behind (neither, I imagine, will they be too envious of the enhanced version's enhanced price). But at least it shows that some companies are finally starting to produce software that really makes use of the 128's full potential.





ZX COMPUTING

FULL CREDIT FACILITIES FOR ACCESS AND BARCLAYCARD.

Lineage: 48p per word. (VAT inclusive)

Semi display: £9.50 per single column centimetre + VAT. Ring for information on series bookings/discounts (minimum 2.5cm).

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

01 - 437 0699

Send your requirements to:
NICOLA BATY
ASP Ltd., 1 Golden Square,
London W1

SOFTWARE

WD Software

FOR THE QL:

JOSS — **base £13**

Forget that tedious, time-consuming syntax! Just move the cursor and press SPACE for all your file commands. Cursor keys or your joystick allow you to access microdrives (up to 8) and floppy discs (as many as our interface allows) with up to 150 files on each! Scroll and print directories, COPY, DELETE or PRINT any file, select TV or Monitor mode before LOADing or RUNning any program. You only use the keyboard to set the DATE or label a device when FORMATting. Easy to use with Psion and other software. No silly icons to learn - JOSS will TELL you what it's going to do! Programmer's toolkit and mass copying/printing utilities also supplied. Specify microdrive-only, Microperipheral or CST-compatible disc versions.

Ref QL7 — **base £7**

1300 useful QL references with ARCHIVE 2 search/print program. Too long for just one cartridge, so if you have RefQL5 just pay 2 and extra media cost to update.

Mdv Extension Cable — **£5.50**

Eight inches long, allows addition of extra microdrives to your QL. Twist it to put their slots facing you.

FOR THE QL: SPECTRUM; BBC; ELECTRON

WD Morse Tutor — **base £4**

Written to teach amateurs, now used by professionals too! Absolute beginner, or stretching your speed to 18 wpm, you won't find anything with more helpful features. What else can offer 100 random sentences as well as all the basics? Disc version unsuitable for BBC B+.

FOR THE SPECTRUM:

WorDfinder (Microdrive/disc only) — **base £8**

For CHEATING at crosswords. Finds m-ss-ng letters, solves, anagrms. 13,000 word vocabulary, so too long to share a cartridge. 10-letter word ending in ATE? No problem.

Tradewind — **base £3**

Sailing/trading strategy game with graphic surprises.

Jersey Quest — **base £3**

Text adventure in time. Background of Jersey folklore from Stone Age to Bergerac.

For export:

QL hardware and software from many sources. Ask for list/quotation.

ORDERING ADD COST OF MEDIUM, POSTAGE £1 OUTSIDE EUROPE.
Mdv or 5.25" floppy = £2 3.5" floppy = £4 Cassette = £0

Payment

By ACCESS/Eurocard/MasterCard or STERLING (UK bank cheques, Eurocheques, drafts or International GIRO). To:

WD Software (ZX), Hilltop, St. Mary, Jersery, C.I. tel: (0534) 81392

QL/SPECTRUM UTILITIES! Wide range of business/practical programs available. SAE/IRC for details (state Micro). S.D. Microsystems (ZX) PO Box 24, Hitchin, Herts.

NEW ZX81 SOFTWARE Games, utilities, adventures, books, our ZX81 Users Club and much more. Send SAE to: A.C.V., 1 Foxwell Square, Southfields, Northampton NN3 5AT.

SLOW SCAN TELEVISION

Transmit and receive SSTV Pictures with this super ZX Spectrum programme. No hardware needed, only **£4.95.**

P. GOODRUM
9 Ryston Close,
Downham Market,
Norfolk, PE38 9BD
Tel: 0366 388615

SERVICES

STOP PLAYING GAMES

Use your computer to make money. Turn your hobby into a home-based income. Full and part time opportunities to cash in on this tremendous market. High earnings easily possible. Open to any amateur micro user and gamer. Write for free details.

Westlink Promotions: (CG)
108 George Street
Edinburgh EH2 4LH.

REPAIRS

COMPUTER REPAIRS

We are the experts, having serviced Sinclair computers since the introduction of the ZX80.

Don't waste money on estimates – we repair Sinclair computers at price quoted inclusive parts, labour, postage, VAT, irrespective of fault. No hidden charges.

Repairs guaranteed for 3 months.

Spectrum	£18.75 inc parts
ZX81	£11.50 inc parts
16 KRam	£9.95 inc parts
Microdrive	£15.95 inc parts
Interface 1-11	£18.75 inc parts
Also	
BBC	£22.00 + parts
Electron	£19.95 + parts
XK Memory Expansion Kit	£15.95

Computer Retailers please phone for Special Trade Price.

Call or send with cheque or P.O.
T.V. Services of Cambridge Ltd.
French's Road, Cambridge, CB4 3NP
Tel. 0223 311371

SPECTRUM REPAIRS

£14.95 inclusive of labour parts and **p&p.** Fast, reliable service by qualified engs., average repair **24hrs. 3 months guarantee**. For help or advice ring:

H. S. Computer Services, Unit 2, The Orchard, Warton, Preston, Lancashire PR4 1BE. Tel: (0772) 632686.

SINCLAIR SERVICE CENTRE

- Fully Guaranteed Fast Repair Service
- Approx. 24hr Turnaround
- £15 av cost or send machine with £1.95 (return p&p) for free estimate
- Personal callers welcome

QUANTUM
33 CITY ARCADE, COVENTRY CV1 3HX
Tel: (0203) 24632

SINCLAIR REPAIRS

SPECTRUM/PLUS	£12.00
KEYBOARD	£8.50
INTERFACE/MICRODRIVE	£17.00 each

All prices are fully inclusive of p&p and VAT. Send Cheque with computer only unless power supply suspected. Callers Welcome.

I.T. WESTERN ELECTRONICS
Unit F2A & F3
Avonside Enterprise Park
Newbroughton Road,
Melksham, Wilts
Tele: (0225) 705017

To advertise your repairs & spares service phone 01-437 0699

SCOTLAND'S Nº 1

For home and personal computer repairs
Specially fast Spectrum service!

- *Same day for most faults*
- *1 hour if delivered personally*
- *Open 6 days a week*
- *Free estimates*
- *Upgrades, Membranes and P.S.U.'s*
- *3 mth. warranty on work done*

Also BBC/CBN/ORIC and PERIPHERALS

MICRO-SERV

95 Deerdykes View
Westfield Industrial Area,
Cumbernauld G68 9HN
Scotland
Tel: Cumbernauld (02367) 37110

Trade, schools and club discount given

SOFTWARE

THE RECKONER

A great utility program. Works out most popular horse racing bets, plus greyhound forecast doubles, and up to a twelve fold for the fixed adds (for the bookmakers football pools).
£3.99. Plus **50p** P&P.
Send cheque or P.O. to

Replay Software, Dept. Z,
8 Stevenage Road,
East Ham, London E6 2AX.

NEW ZX81 HI-RES GAME Arcade action with WAR WEB, £3.95. Fun for all the family with POOTER PUZZLER, £2.95. Send S.A.E. for leaflet. Pooter Games, 24 Parsloes Avenue, Dagenham RM9 5NX.

PLAN YOUR ADVERTISING BUDGET WITH ZX COMPUTING. RING 01-437 0699 NOW

ACCESSORIES

SPECTRUM RESET SWITCH - Prevents damage to your computer when power plug pulled in and out. **Only £3.49 inclusive**. Cheque or P.O. to Central Electronics, St Andrews St, Greenock PA15 1HG. Other accessories. Write for free list.

WARNING NOTICE

Advertisements placed in this magazine are to be in strict compliance with our standard conditions (copies of which conditions are available on request) and on the clear understanding that the advertiser warrants that his advertisement(s) does not infringe any copyright or condition of sale of any interested party in the advertised product.

Further, the advertiser indemnifies the proprietors of this magazine in respect of costs, damages, or any other claims brought against them as a result of legal action arising from the publication of the advertisement.

Any breach of these terms or the said conditions may result in prosecution of the advertiser by the proprietors.

THE PRESS

The Press
Gilsoft
£6.95

The press is an adventure utility to complement the ever-popular Quill and Illustrator programs. So don't expect this review to be technical. I firmly believe adventures should be written not by programmers, but by authors, who have little need for technical jargon.

Primary feature of The Press is the text-compressor, which reduces the amount of memory used up by location description and messages in your Quilled game. The program is loaded in from The Quill while your database is present. A short menu appears, the new feature of which is the 'Compile and Compress' option. On selection of this, you are given the choice of fast or slow compression. Fast will probably take around an hour, while Slow can take ten. You are also given the choice of selecting a bias for mostly message or location compression, or an even split.

Does the compression work? I tried The Press on a fairly standard text only game that used all but two bytes of the Quill's standard approximately 30K free memory. On fast compression, I managed to free 9677 bytes, and the process took just half an hour. Slow compression was not noticeably better, saving 9936 bytes but taking eight hours to do so. However, with more complex and varied prose, and careful bias selection, slow should be more impressive (Gilsoft say 50% is possible on some texts). Most people — particularly commercial writers — will always use Slow I expect, unless the compression is being carried out to cram in just a little that couldn't previously be fitted in. There is an option to use a compressor 'dictionary' prepared with a previous adventure; this is faster than normal 'fast', though to most people memory will be more important than speed.

The compressor has useful implications. You can now write a full length text adventure, compress it and add previously impossible graphics. Or you can produce relatively massive text adventures by compressing, adding more with The Quill, recompressing etc. Using The Expander on the cassette's reverse, you can make even larger games by using memory normally taken up with the main Quill program. However, you cannot add new locations or messages with it, you can only 'amend' them. You must include blank files in your original which can be filled in after compression so your adventure should be precisely planned.

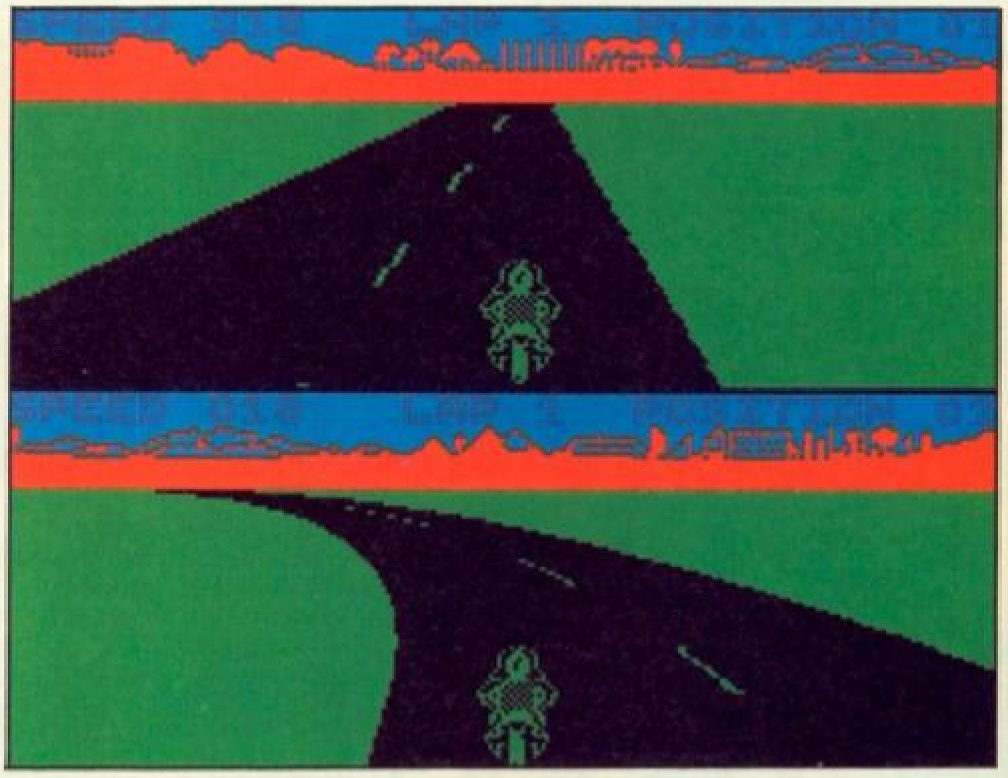
The Press includes numerous other helpful additions to The Quill, most of which previously formed 'The Patch'. The functions, which are controlled using a flag and a PAUSE statement in your original Quill program, include; split screen graphics of any size you wish (which they scroll up with the text), single command graphics on/off control, a single command restart, some sound effects, RAM SAVE/LOAD facility, the ability to incorporate different typefaces, and a few features which make including your own routines and loading game data between parts of a multi-part adventure easier. All of these are extremely useful and most are vital to use if you wish to market your adventure.

If you're already a user of The Quill, this package is a powerful and easy to use expansion. However, if you intend to write commercially, or you don't have the Quill yet, wait until Gilsoft launch their new, Professional Adventure Writer.

SPEED KING 2

A high calibre motorcycle race simulation from Mastertronic

Speed King
Mastertronic (MAD Games)
£2.99



If you want to add a motorcycle simulation game to your software collection, look no further. Speed King II is an excellent package, crammed with options that for a budget price offers outstanding value for money.

For starters there are nine tracks to choose from, ranging from Silverstone (the easiest) to Brands Hatch (the hardest). The race action itself has been pitched at just the right degree of difficulty. You start at the back of the grid with 19 other riders to overtake on your way to the finishing line. It's impossible to crash; hit another rider and your speed plummets to zero as it does when you career off the track. While lacking in realism this makes for a better game as there's nothing more annoying than being eliminated for a tiny mistake.

The handling of the bike (there are keyboard, Interface 2 and Kempston options) is very responsive and unlike some motorcycle games does not require pinpoint accuracy on every turn; missing the optimum line or cornering too fast will just result in a rapid drop down the field.

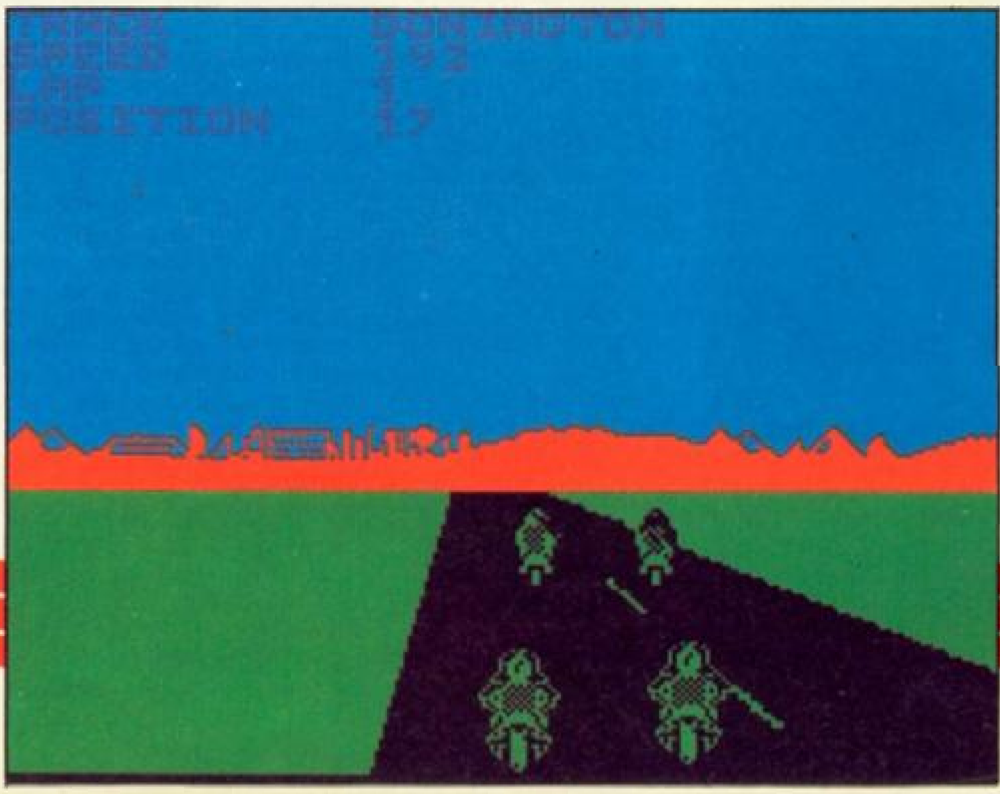
The number of laps can be varied from 1 to 9 and as well as the one player game there's also a two-player option with a split screen display.

With such a range of options available, Speed King II will undoubtedly give hours of racing pleasure because if you find the track too simple you can go on to a harder one or alternatively cut down the number of laps.

To keep track of your progress there is an after race display which gives your placing, best placing so far, your fastest lap and the current lap record.

For the race game fanatic Speed King II can be highly recommended.

MONSTER HIT




COMPUTER
GAMER
It's Supermag!
SAME GREAT MAG...
GREAT NEW LOOK!!
FEBRUARY ISSUE ON SALE 23 JANUARY


AMSTRAD 8.95 COMMODORE 8.95 SPECTRUM 7.95
It was a one-in-a-million accident – but Number Five, designed to be a strategic artificially intelligent weapons system, the most sophisticated robot on the planet, has escaped – and has come to the conclusion that he's alive! Now the scientist who put him together wants to take him apart
SHORT CIRCUIT TM
again to find out what went wrong. The president of Nova Robotics wants to capture him before the weapons he's carrying kill millions of civilians. And the security chief wants to blow him up so that he can get home in time for dinner.
YOU are Number Five...YOU are alive and YOU have got to stay that way!
ocean
Ocean Software Limited
Ocean House · 6 Central Street · Manchester · M2 5NS
Telephone 061 832 6633 · Telex 669977 Oceans G
Short Circuit is a trademark of Tri-Star Pictures, Inc. and PSO Presentations.
© 1986 Tri-Star Pictures, Inc. and PSO Presentations. All Rights Reserved.